Learning detector ensembles for object detection

Zhou, Chunluan

2018

Zhou, C. (2018). Learning detector ensembles for object detection. Doctoral thesis,
Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/88826>

<https://doi.org/10.32657/10220/46010>

**NANYANG
TECHNOLOGICAL
UNIVERSITY**

**LEARNING DETECTOR ENSEMBLES
FOR OBJECT DETECTION**

**ZHOU CHUNLUAN
SCHOOL OF ELECTRICAL & ELECTRONIC
ENGINEERING
2018**

LEARNING DETECTOR ENSEMBLES FOR OBJECT DETECTION

ZHOU CHUNLUAN

School of Electrical & Electronic Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirement for the degree of
Doctor of Philosophy

2018

Acknowledgements

First of all, I would like to thank my main supervisor Prof. Junsong Yuan for his patient guidance in my research work throughout my PhD study. Under his guidance, I have gotten a deep understanding of how to do research and realized the importance of communication and collaboration. He also allowed me sufficient freedom to explore research topics I was interested in and provided insightful suggestions. I was transferred to Prof. Kai-Kuang Ma in the last half year. I would like to thank Prof. Ma for the discussion and guidance on the thesis revision and oral defence.

I thank members of our research group at Nanyang Technological University: Gangqiang Zhao, Gang Yu, Hongxing Wang, Ye Luo, Yang Xiao, Jianyu Yang, Yuwei Wu, Shizheng Wang, Jianfeng Ren, Tran Du, Yuning Jiang, Zhou Ren, Chaoqun Weng, Bo Hu, Hui Liang, Kang Dang, Das Bhattacharjee Sreyasee, Anirban Chakraborty, Jiong Yang, Junwu Weng, Jingheng Luo, Mengjia Yan, Jiaqi Zhang, Tan Yu, Liuhao Ge, Zhenzhen Wang, Weixiang Hong, Suchen Wang and Zhigang Tu. We have had stimulating discussions through group meetings and personal communications and also had a lot of fun in the past five years.

I want to thank my parents for their support, encouragement and love through all the time of my study which have motivated me to work hard and lead a happy life.

Table of Contents

Acknowledgements	i
Table of Contents	iii
Abstract	vii
List of Figures	ix
List of Tables	xiii
1 Introduction	1
1.1 Object Detection	1
1.2 Challenges	2
1.3 Detector Ensembles	4
1.4 Contributions	5
1.5 Thesis Organization	8
2 Related work	9
2.1 Deformable Part Models	9
2.2 Occlusion Handling for Generic Object Detection	10
2.3 Approaches using Deep Learning for Generic Object Detection	12
2.4 Pedestrian Detection	12
2.5 Occlusion Handling for Pedestrian Detection	13
3 Non-rectangular Part Discovery for Generic Object Detection	15
3.1 Introduction	15
3.2 Non-rectangular Part Discovery	18
3.2.1 Deformable Part-based Model	18
3.2.2 Part Initialization by Greedy Search	20
3.2.3 Our Approach	21
3.2.3.1 Formulation	21
3.2.3.2 Optimization	24
3.3 Experiments	26
3.3.1 Implementation Details	26
3.3.2 Parameter Selection	28
3.3.3 Results on 20 Object Categories	28

3.4	Conclusion	29
4	Occlusion Pattern Discovery for Generic Object Detection	33
4.1	Introduction	33
4.2	Occlusion Pattern Discovery	37
4.2.1	Generating candidate occlusion patterns	37
4.2.2	Selecting occlusion patterns	39
4.2.3	Extending to multiple viewpoints	42
4.3	Object detection with discovered occlusion patterns	44
4.3.1	Modeling occluders	44
4.3.2	Learning model parameters	45
4.3.3	Object detection with occlusion inferring	46
4.3.3.1	Sliding-window search	46
4.3.3.2	Combination with another detector	46
4.4	Experiments	47
4.4.1	Single-viewpoint occlusion pattern discovery	47
4.4.1.1	Implementation	47
4.4.1.2	Experimental results	48
4.4.2	Multiple-viewpoint occlusion pattern discovery	50
4.4.2.1	Implementation	50
4.4.2.2	Car detection results	51
4.4.2.3	Performance analysis on partially occluded objects	52
4.4.2.4	Results with varying numbers of occlusion patterns	54
4.4.2.5	Results on 20 object categories	54
4.4.3	Experiments with Faster R-CNN	58
4.4.3.1	Implementation	58
4.4.3.2	Experimental results	58
4.5	Conclusion	59
5	Learning to Integrate Part Detectors for Pedestrian Detection	63
5.1	Introduction	63
5.2	Training Part Detectors	65
5.3	Integrating Part Detectors	68
5.3.1	Linear Integration	69
5.3.2	Our Approach	70
5.4	Experiments	72
5.4.1	Implementation Details	72
5.4.2	Results of Part Detectors	73
5.4.3	Results of Different Integration Methods	74
5.5	Conclusion	79
6	Multi-label Learning of Part Detectors for Pedestrian Detection	83
6.1	Introduction	83
6.2	Multi-label Learning of Part Detectors	86
6.2.1	Part representation	86

6.2.2	Multi-label formulation	86
6.2.3	Learning via boosting	89
6.2.4	Complexity analysis	92
6.3	Occlusion handling with part detectors	92
6.3.1	Integrating part detectors	92
6.3.2	Contextual rescoring	94
6.4	Experiments	96
6.4.1	Experiments with channel features	97
6.4.1.1	Implementation	97
6.4.1.2	Experimental results on Caltech	97
6.4.1.3	Experimental results on CUHK	103
6.4.2	Experiments with CNN features	105
6.4.2.1	Implementation	105
6.4.2.2	Detection speed	105
6.4.2.3	Experimental results on Caltech	106
6.4.2.4	Experimental results on CUHK	109
6.5	Conclusion	110
7	Conclusion	113
	Publications	115
	Bibliography	117

Abstract

Telling "what is where", object detection is a fundamental problem in computer vision and has a broad range of applications such as video surveillance and autonomous driving. One major challenge of object detection comes from large intra-category appearance variations which are caused by factors including subcategory, viewpoint, deformation and occlusion. Large appearance variations make it difficult to model an object category properly such that the object category is well distinguished from other object categories as well as backgrounds. Learning a detector ensemble is a widely adopted solution to appearance variation handling. Appearance variations resulting from subcategories and viewpoints are usually handled by clustering object examples of a category into groups each of which represents one subcategory or viewpoint and then learning a detector for each group. For dealing with appearance variations due to deformations and occlusions, part-based detection methods have demonstrated their promise by integrating a set of part detectors to form a part detector ensemble. This thesis studies how to learn detector ensembles to better address deformations and occlusions, particularly the latter situation, for generic object detection as well as pedestrian detection.

For generic object detection, two approaches are proposed to handle deformations and occlusions respectively based on a classic part detector ensemble, deformable part model (DPM). The former discovers a set of non-rectangular parts which can well fit object structures to replace the original rectangular parts in the DPM. The discovered non-rectangular parts can better capture the appearance of local regions and structural deformations of objects. The latter discovers a set of representative and discriminative occlusion patterns which share the same set of parts from a DPM trained on fully visible object examples. The discovered occlusion patterns are themselves DPMs, and when properly tuned, can be applied directly or combined with state-of-the-art detectors, e.g. Faster R-CNN for improving detection performance and achieving part-level occlusion reasoning.

For pedestrian detection, two approaches are developed to improve two modules of a commonly used framework of learning a part detector ensemble respectively for handling occlusions. The

first approach focuses on how to integrate part detectors properly to reduce negative effects from unreliable and irrelevant part detectors on heavily occluded pedestrian detection. The second approach aims to learn reliable part detectors jointly by sharing a set of decision trees among the part detectors to exploit part correlations and also reduce the computational cost of applying these part detectors. Experimental results on pedestrian detection benchmark datasets show promising performance of the two approaches for detecting partially occluded pedestrians, especially heavily occluded ones.

List of Figures

1.1	Example images from PASCAL VOC2007 dataset [1] with object annotations. .	2
1.2	Appearance variations caused by subcategory, viewpoint, deformation and occlusion.	4
3.1	Rectangular parts vs non-rectangular parts. (a) Part configuration discovered by the greedy search approach. (c,e,g) Detection examples of the part configuration (a). Red bounding boxes denotes the matching regions of the root filter. (b) Part configuration discovered by our approach. (d,f,h) Detection examples of the part configuration (f). It can be seen that the actual shapes of bicycle seat and handle are better fitted by non-rectangular parts and the part configuration (b) better matches the structure of a left/right facing bicycle.	17
3.2	Toy example of part filter initialization. (a) An object example consisting of two parts. (b) A deformed object example whose two parts move away from each other. (c) The enlarged root filter with two 3×3 highest-energy part filters obtained by the greedy method. Bright cells indicate high-energy regions and dark cells indicate low-energy regions. (d) A better part-filter configuration which agrees with the structure of the two object examples.	21
3.3	Our part discovery method. (a) The root filter of right-facing bicycles. (b) The average score map of cells. (c) The initial part-filter configuration Λ_0 . The black grids denote the cells which are removed from the enlarged root filter. (d) The final part-filter configuration after refinement. For clarity, $\tau = 0$ is used in this example to obtain the part-filter configuration without overlap.	25
3.4	Bicycle model of 6 components. For each component, the root filter is show to the left of the part filters which are displayed by color polygons.	27
3.5	Detection examples of our approach for the 20 object categories in the PASCAL VOC2007 dataset.	30
4.1	Overview of the proposed approach to occlusion pattern discovery applied to the person category.	34
4.2	Occlusion reasoning. The red bounding box in (a) is a candidate detection which is obtained by Faster R-CNN. The candidate detection is matched to the occlusion pattern shown in (b). Green and blue bounding boxes in (a) show visible parts and the estimated full human body respectively after the candidate detection is aligned with the occlusion pattern.	36

4.3	Example of occlusion pattern. (a) The occlusion pattern has two visible parts marked by blue and green rectangles respectively. (b) All weights in the invisible portion of the root filter are equal to 0. (c) Weights in the visible portion (the top area) are set to 0 and cells close to the boundary of the visible region have large weights. (d) The occluder filter of the occlusion pattern has zero weights in the visible portion.	38
4.4	Comparison of different approaches using HOG features on the person category in VOC2007.	48
4.5	Occlusion patterns of GRAM [2] (Column 1) and our approach (Columns 2 to 7).	49
4.6	Person detection examples. Columns 1-4 show some examples which are matched with correct occlusion patterns. Columns 5-6 show some failure examples which are matched with wrong occlusion patterns. For clarity, only one detection is displayed in each image.	49
4.7	Comparison of different approaches using HOG features on the car category in VOC2007.	51
4.8	Results of DPM and OCCL-D on four occlusion levels. Black dashed lines indicate overall AP_N . Red bars indicate standard errors. The horizontal axis represents different occlusion levels: N = None, L = Low, M = Moderate and H = High.	53
4.9	Results of OCCL-D on the car and dog categories in the VOC2007 dataset. PK means K occlusion patterns are used. Numbers in the legend refer to average precisions.	55
4.10	Examples of discovered occlusion patterns and detections. The regions of a part in a detection and its corresponding occlusion pattern are marked with the same color. For clarity, only one typical detection is displayed in each image.	57
4.11	Detections with occlusion inferring. Red bounding boxes are detections obtained by Faster R-CNN. These detections are aligned with their matching occlusion patterns. Green and blue bounding boxes represent visible parts and estimated full objects respectively. Some images are padded with black regions to display estimated full objects. For clarity, only one typical detection is displayed in each image.	60
5.1	Different integrations of part detectors. Left: The average pattern 0 is obtained by averaging visible parts of pedestrian examples in the Caltech dataset [3] and the top 4 occlusion patterns whose detectors have the highest weights are shown below. Blue regions in the occlusion patterns indicate visible parts of the human body. The pedestrian which is heavily occluded in the image below is ranked at 4th among top 5 regions. Right: Our approach clusters the pedestrian examples into 2 groups each of which selects its own compatible part detectors. The detectors associated with the average pattern 1 rank the pedestrian in the image below at 1st.	64
5.2	Examples of occlusion patterns. (a) The right portion of the human body is occluded. Two instances marked by red rectangles are shown on the right: one is occluded by another pedestrian and the other is truncated by the image boundary. (b) The lower body is occluded. Two pedestrians on the right are occluded by cars.	66

5.3	Process of training a detector for the upper-body pattern.	67
5.4	Visible part clustering. (a) Red and green bounding boxes show the full body and visible part of a pedestrian example respectively. (b) Four cluster centers obtained by K-means on pedestrian examples in the Caltech dataset. Cyan and white regions indicate visible and occluded parts respectively.	70
5.5	(a) Results of part detectors. (b) Results of our LDCF implementation and some state-of-the-art approaches using channel features.	73
5.6	Common occlusion patterns.	74
5.7	Top 20 occlusion patterns of Linear with associated combination weights.	75
5.8	Average occlusion patterns and top occlusion patterns of two groups of Ours-g2.	76
5.9	Results of our integration approach and three baselines.	78
5.10	Results of our integration approach with different numbers of groups.	79
5.11	Results of four integration approaches and some state-of-the-art approaches. Solid lines represent channel-feature based approaches and dashed lines represent deep-learning based approaches.	80
5.12	Results of our approach and new state-of-the-art approaches.	81
6.1	Occlusion handling. (a) Top five detections of a full-body detector. The heavily occluded pedestrian (Blue bounding box) is only ranked at 5th. (b) Detection scores of 20 part detectors on the five detections in (a). The first detector is the full-body detector. Figure 6.2 shows the 20 parts. (c) The five detections in (a) are re-ranked by properly integrating the 20 part detectors. The heavily occluded pedestrian (Blue bounding box) is ranked at 2nd.	84
6.2	Part pool. Blue regions denote parts. The first part is the whole body which is modeled as a template of 6×3 cells. Parts 2 to 17 are designed to handle situations where occlusions occur from the left, right or bottom and the last three parts are used for detecting the lower body.	87
6.3	Example labeling. The blue and red bounding boxes are the visible portion and full body of the pedestrian example respectively. The green bounding box is the standardized full body and the yellow bounding box is the new visible portion inside the standardized full body. The purple bounding box shows the image region of the upper-body part on the pedestrian example. The cost vector is calculated based on the purple and yellow bounding boxes.	88
6.4	Part scores on top 5 scoring image regions. The number above each region denotes its ranking. For the fully visible pedestrian (Red bounding box), all part detectors give high detection scores consistently (Red curve). For the partially occluded pedestrian (Blue bounding box), only the detectors of visible parts (e.g. P3, P4, P5, P9, P10, P14 and P15) output high detection scores (Blue curve). Background regions (Green, yellow and cyan bounding boxes) receive relatively low scores from all the part detectors (Green, yellow and cyan curves).	93
6.5	Detection distribution. Red bounding boxes are thirty detections with the highest scores. Detections are densely located around the two pedestrians marked by green bounding boxes. Detections are sparsely distributed in other regions.	95
6.6	Results of part detectors using different part representations on the Caltech dataset.	98
6.7	Two part pools.	102

6.8	Comparison with state-of-the-art approaches using channel features on the Caltech dataset.	103
6.9	Results using channel features on the CUHK dataset.	104
6.10	Comparison with state-of-the-art approaches using CNN features on the Caltech dataset.	107
6.11	Comparison with new state-of-the-art approaches on the Caltech dataset.	108
6.12	Detection examples of our approach on the Caltech dataset.	109
6.13	Results on CUHK using CNN features.	110
6.14	Detection examples of our approach on the CUHK dataset.	111

List of Tables

3.1	Average precision (AP) for 10 object categories in the PASCAL VOC2007 dataset with different λ .	28
3.2	Performance comparison using Average Precision (AP) on 20 object categories in the PASCAL VOC2007 dataset.	29
3.3	Performance comparison using Average Precision (AP) on 20 object categories in the PASCAL VOC2010 dataset.	31
4.1	Summary of variables for occlusion pattern discovery.	43
4.2	Average Normalized Precisions of DPM and OCCL-D on different occlusion levels. The All level includes all occlusion levels.	53
4.3	Average precision of different approaches using HOG features on 20 object categories in the VOC2007 dataset. DPM-BB uses bounding-box prediction to refine the results of DPM. Both OCCL and OCCL-D do not use any post-processing.	56
4.4	Average precision of different approaches using HOG features on 20 object categories in the VOC2010 dataset. DPM-BB uses bounding-box prediction to refine the results of DPM. Both OCCL and OCCL-D do not use any post-processing.	56
4.5	Average precisions of Faster R-CNN and our approach on 20 object categories in the VOC2007 dataset. ResNet-101 is used for Faster R-CNN.	59
4.6	Average precisions of Faster R-CNN and our approach on 20 object categories in the VOC2010 dataset. ResNet-101 is used for Faster R-CNN.	59
5.1	Results of different integration methods. The bold number in each row indicates the best performance of the corresponding method.	77
6.1	Comparison of separate learning (SL) and joint learning (JL) using channel features on the Caltech dataset. P1-P4, P6 and P11 are six typical parts shown in Fig. 6.2. The last row shows the average improvements on the three subsets brought by joint learning.	99
6.2	Comparison of cost-sensitive labeling and hard learning using channel features on the Caltech dataset. P1-P4, P6 and P11 are six typical parts shown in Figure. 6.2. The last row shows the average improvements on the three subsets brought by cost-sensitive labeling.	99
6.3	Results of full-body detectors and integration approaches using channel features on the Caltech dataset.	100
6.4	Results of approaches using different part pools on the Caltech dataset.	101

6.5	Results of full-body detectors and integration approaches using CNN features on the Caltech dataset.	107
-----	--	-----

Chapter 1

Introduction

1.1 Object Detection

Object detection is a fundamental problem in computer vision. Given an image, the goal of object detection is to localize objects of interest in the image. Specifically, for the given image, the output of an object detection algorithm is a set of $\langle \text{region}, \text{category} \rangle$ pairs. Each $\langle \text{region}, \text{category} \rangle$ pair specifies the region of an object in the image and the category which the object belongs to. The region of an object is usually annotated by a rectangular bounding box that tightly encloses the visual content of the object. Figure 1.1 gives some examples of object annotations.

Object detection has many applications. Face detection has become a common important module in a digital camera. An autonomous car needs to detect objects like people and vehicles in front of it in order to avoid collisions. In a traffic surveillance system, it is usually required to detect vehicles and pedestrians in order to analyze traffic conditions or give traffic alerts. In addition, object detection can provide useful information for other computer vision tasks, such as object tracking [4], activity recognition [5], and automated image captioning [6].

Generally speaking, object detection consists of three components: candidate region sampling, region representation and region classification. The first component samples a set of candidate

FIGURE 1.1. Example images from PASCAL VOC2007 dataset [1] with object annotations.

regions in an image which may contain objects of interest. These candidate regions can be obtained by window sliding (*e.g.* [7]) or proposal generation (*e.g.* [8]). Window sliding samples regions uniformly in the image at different locations and scales, while proposal generation samples regions nonuniformly, giving priority to regions which are more likely to contain objects of interest. The candidate regions generated by this component are passed to the next two components for further processing. The second component extracts features from these candidate regions to represent them. Commonly used features for generic object detection include bag of words [9], histograms of oriented gradients [10] and features learned by a deep neural network [8]. Specific features are also designed for some object categories, *e.g.* haar features for face detection [7] and channel features for pedestrian detection [11]. The third component learns a classifier to categorize these regions based on their feature representations. For object detection, support vector machine (SVM) [10, 12], Boosting [7, 11, 13] and deep learning [8, 14, 15] are typical methods for learning classifiers.

1.2 Challenges

An object category usually exhibits large appearance variations, which makes it challenging to detect objects belonging to the category. Factors that cause large appearance variations within an object category include subcategory, viewpoint, deformation and occlusion.

Subcategory. Some object categories consist of subcategories which have different structures, resulting in large intra-category variations in appearance. For example, the boat category can be divided into barge, canoe, lifeboat, sailboat, ferry and other sub-categories. The appearance of one boat sub-category may be very different from that of another, *e.g.* ferry and lifeboat. Figure 1.2(a) shows some object examples of different boat subcategories.

Viewpoint. An 2D object in an image is obtained by photographing its 3D physical form in the real world from a specific viewpoint. The appearance of an object, when photographed from different viewpoints, may change drastically. For example, the shape of a bicycle with a frontal view is visually very dissimilar to that of the same bicycle with a side view. When objects of a category to be detected have different views, they would possibly exhibit large appearance variations. Figure 1.2(b) shows some objects of the bicycle category photographed from different viewpoints.

Deformation. Deformation causes objects to change their shapes. A balloon deforms due to stretching, bending, expanding, shrinking, etc, exhibiting all kinds of shapes. For an object consisting of parts which can move relative to each other, part movements lead to different part configurations, causing the shape of the object to change. Take the cat category for instance. Cats can deform flexibly (*e.g.* part deformation and part movement), which results in large appearance variations within the cat category as shown in Figure 1.2(c).

Occlusion. Occlusions occur frequently in practical applications. For example, for pedestrian detection in a street scene, a pedestrian is often occluded by a car or another pedestrian. Besides being occluded by another object, an object can also be occluded by itself, *e.g.* when one of its parts moves in front of another. When different parts of an object are occluded, its appearance changes. Figure 1.2(d) shows some person examples which look visually dissimilar due to partial occlusions.

(a) Subcategory

(b) Viewpoint

(c) Deformation

(d) Occlusion

FIGURE 1.2. Appearance variations caused by subcategory, viewpoint, deformation and occlusion.

1.3 Detector Ensembles

Large appearance variations within an object category make it difficult to distinguish objects of the category from objects of other categories and background regions. Learning an ensemble of detectors is a widely adopted solution for handling large appearance variations [2, 10, 16–26]. A detector ensemble is a set of detectors which are trained and combined to detect instances of an object category. Generally, individual detectors in the ensemble are trained to capture

different characteristics of the object category, *e.g.* viewpoints and parts. This can reduce the appearance variation each individual detector needs to handle. Then, these individual detectors are properly combined to detect various instances of the object category which exhibit large appearance variations. Appearance variations resulting from subcategories and viewpoints are usually handled by clustering objects of a category into groups each of which represents one subcategory or viewpoint and then learning a detector for each group [10, 18–21]. Since each detector handles a subset of instances of the object category, a final decision is usually made in a winner-take-all fashion. Part-based detection methods [2, 10, 16, 22–30] have demonstrated their promise by integrating a set of part detectors to deal with appearance variations caused by deformations and occlusions. Because part detectors are usually less discriminative than a holistic detector, the responses of part detectors and relations between parts are usually exploited together to make a final decision. Both training and combining the individual detectors are important for constructing a detector ensemble. First, the set of individual detectors should be diverse such that they cover important characteristics of an object category. Second, the quality of individual detectors is important. The individual detectors should be as discriminative as possible to reduce the chance of introducing noise when they are combined. Third, the individual detectors should be carefully combined to well distinguish objects of different categories as well as background regions. In this thesis, we study how to exploit detector ensembles to handle large appearance variations for general object detection as well as pedestrian detection. In Chapters 3, 4 and 6, we explore how to train individual detectors for a detector ensemble to solve different problems in object detection, while in Chapter 5, we study how to combine part detectors to handle occlusions for pedestrian detection.

1.4 Contributions

This thesis studies how to learning detector ensembles to better address deformations and occlusions, particularly the latter situation. The contributions of this thesis are summarized as follows.

Non-rectangular part discovery for generic object detection. Generally, an object is comprised of a set of parts. Part appearance and structural relation among parts provide important cues for recognizing objects. A deformable part model (DPM) [10] is specially designed to capture the appearance of parts and structural deformations of objects. The DPM is a typical part detector ensemble which consist of a root which represents an entire object and several parts which model the appearance of local regions and can move relatively to the root to capture structural deformations. Parts play an important role in the DPM. However, little work has been proposed to discover parts for the DPM. Most DPM-based approaches adopt a heuristic method [10] to obtain a number of rectangular parts, which may not be optimal for some object categories. In Chapter 3, we present a novel approach to discover a set of non-rectangular parts that can better fit object structures. Replacing the rectangular parts with the non-rectangular parts discovered by our approach helps to improve the performance for the DPM.

Occlusion pattern discovery for generic object detection. Occlusions occur frequently in practical applications, which presents a great challenge for object detection. A common strategy for handling occlusions is to learn a set of occlusion-specific classifiers each of which corresponds to an occlusion situation. One key step for this strategy is to discover a set of occlusion patterns which well cover possible occlusion situations. In Chapter 4, we present a data-driven approach to discover occlusion patterns based on the part detector ensemble DPM. Our approach first trains a DPM on fully observed object examples. Each occlusion pattern contains only a subset of visible parts, thus the total number of occlusion patterns is exponential to the number of parts, i.e. m parts will generate 2^m occlusion patterns to compose an occlusion pattern pool. From this occlusion pattern pool, we look for a small group of occlusion patterns that are: (1) representative patterns that can well explain training examples, and (2) discriminative patterns that have high detection performance individually. To select such occlusion patterns, we formulate occlusion pattern discovery as a facility location problem, which can be solved effectively by greedy search. The discovered occlusion patterns are themselves DPMs and can be used as object detectors when properly tuned. They can also be combined with state-of-the-art detectors (e.g. Faster R-CNN) for improving detection performance and

achieving part-level occlusion reasoning.

Integrating part detectors for pedestrian detection. It is a challenging problem to detect partially occluded pedestrians due to the diversity of occlusion patterns. Although learning an ensemble of part detectors can help handle various partial occlusions [31, 32], it is a non-trivial problem to integrate these part detectors properly. A direct combination of all part detectors can be affected by unreliable detectors and usually does not favor heavily occluded pedestrian examples, which can only be recognized by few detectors. In Chapter 5, we present a new approach to integrate part detectors for heavily occluded pedestrian detection. Instead of combining all part detectors into a generic detector for all occlusions, we categorize occlusions based on how pedestrian examples are occluded into K groups. Each occlusion group selects its own part detectors and fuses them linearly to obtain a classifier. An L1-norm linear support vector machine (SVM) is adopted to select and fuse part detectors for the K classifiers simultaneously. Thanks to the L1-norm linear SVM, unreliable and irrelevant detectors are removed for each group. Experimental results show promising performance of our approach for detecting heavily occluded pedestrians.

Multi-label learning of part detectors for pedestrian detection. For the framework of learning an ensemble of part detectors to handle occlusions for pedestrian detection, part detectors are usually learned separately [31–36], which ignores part correlations. The computational cost of applying separately learned part detectors increases linearly with the number of parts. In Chapter 6, we propose a multi-label learning approach to jointly learn part detectors to capture partial occlusion patterns. The part detectors share a set of decision trees which are learned and combined via boosting to exploit part correlations and also reduce the computational cost of applying these part detectors for pedestrian detection. The learned decision trees capture the overall distribution of all the parts. When used as a pedestrian detector individually, our part detectors learned jointly show better performance than their counterparts learned separately in different occlusion situations. For occlusion handling, several methods are explored to integrate the part detectors learned by the proposed approach. Context is also exploited to further improve the performance. The proposed approach is applied to hand-crafted channel features

and features learned by a deep convolutional neural network, respectively. Experiments on two pedestrian detection datasets show state-of-the-art performance of our approach for detecting occluded pedestrians, especially heavily occluded ones.

1.5 Thesis Organization

The remainder of this thesis is organized as follows. Related work is discussed in Chapter 2. In Chapters 3 and 4, two approaches are proposed to handle deformations and occlusions respectively for generic object detection based on a deformable part model. In Chapters 5 and 6, two approaches are proposed to improve two modules of a commonly used framework of learning a part detector ensemble respectively to handle occlusions for pedestrian detection. Chapter 7 concludes the whole thesis.

Chapter 2

Related work

Object detection is a fundamental problem in computer vision and has received much attention in past few decades [3, 7–10, 12, 14, 37, 38]. Many approaches have been widely adopted to address appearance variations for object detection, including sub-category learning [18–20, 39], deformable part models [10, 26, 39–43] and occlusion handling [2, 22–25, 31, 33, 44, 45]. Below we summarize some works which are most related to the contributions of this thesis. We also review some approaches based on deep learning which dominate state-of-the-art performance for object detection.

2.1 Deformable Part Models

Objects of some categories can deform flexibly, which increases intra-class appearance variations. The deformable part model (DPM) [10] is a typical part detector ensemble and is effective to capture object deformations. Many approaches have been proposed to improve it. Several approaches combine different features, such as color attributes [46], scale-invariant feature transform (SIFT) [40], local binary pattern (LBP) [41] and segmentation features [47], with histogram of oriented gradients (HOG) [12] to improve the discriminative power of the DPM. In [43], HOG is replaced by histograms of sparse codes (HSC) in the DPM, achieving a significant performance gain. Some sub-category learning methods [18, 22, 48–51] are proposed

to learn better components for the DPM. Flexible variants [22, 52] of the DPM are introduced to handle partial occlusions with part-level supervision for training. For detection, part filters in the two variants can be turned on or off automatically to better match a partially occluded object. In [53, 54], contextual information is exploited to improve the detection performance for the DPM. Category-specific information like texture [55] and irregular patches [56] is used for refining the results obtained by the DPM. Segmentation-aware DPM [57] is proposed to remove noise in HOG representations of objects for the DPM. In [58], score calibration and model training are incorporated into an expectation maximization (EM) framework to obtain a better model. The performance of the DPM is significantly improved by implementing it using a convolutional neural network with HOG replaced by convolutional features [59]. In [60, 61], the DPM is extended from 2D to 3D for 3D object detection.

Due to its popularity, some approaches have been proposed to accelerate the detection process for the DPM. The detection process for a single object category is accelerated by employing KD-Ferns [62], leveraging transform [42], and adopting the cascade [63] or coarse-to-fine [64] detection framework. In [65], cascade detection, filter decomposition and lookup tables are exploited to greatly improve the detection speed of the DPM. Real-time detection is achieved by using a variety of mechanisms like cascade detection and feature interpolation and compression with a little loss of accuracy [66]. In [67], locality-sensitive hashing is exploited to efficiently detect a large number of object categories on a single machine. Building shared intermediate representations for part filters [68, 69] is another way to accelerate the DPM for multiple object categories. In [70], both training and testing is speed up for the DPM with decorrelated features.

2.2 Occlusion Handling for Generic Object Detection

Occlusions occur frequently and are another major source for large appearance variations. Many approaches have been proposed to improve performance for detecting partially occluded objects. Approaches to occlusion handling for object detection can be mainly categorized into two types. The first type uses a single classifier which is able to both distinguish objects of

interest from background regions and infer occlusions [71–73]. Given a candidate region, the classifier infers the visible portion of a potential object in the candidate region and then scores it based on the visible portion by the same classifier [71, 73] or an additionally trained classifier [72]. The second type learns a set of occlusion-specific classifiers each of which handles one or more occlusion situations [2, 22–25, 31, 44, 45, 52, 74]. A candidate region is scored by properly integrating outputs from the occlusion-specific classifiers. Approaches belonging to this type usually require a set of occlusion patterns for learning occlusion-specific classifiers [2, 23–25, 31, 44, 45, 74]. In [2, 31], occlusion patterns are manually designed according to prior knowledge. 3D object models from computer-aided design (CAD) are used to generate occlusion configurations from which occlusion patterns are obtained by representative selection [23] or clustering [25]. In [74], 3D CAD models are projected to match appearance of 2D objects and occlusion patterns are obtained by selecting a set of representatives from an occlusion pattern pool. In [24, 44, 45], occlusion patterns involving one or two objects of the same category are discovered by considering aspect ratios of bounding-box annotations and relative locations of adjacent bounding-box annotations. Occlusion patterns of multiple objects can be helpful when objects of the same category often appear in a group and occlude each other. Two flexible variants [22, 52] of the DPM are proposed for handling occlusions. Parts in the two DPM variants can be switched on or off parts to adaptively match the visible region of an object. Both DPM variants do not need to obtain occlusion patterns but require part-level annotations which are difficult to obtain for training, restricting its scalability to large datasets. In addition, Poselets [75, 76] and other hough-voting based approaches [77, 78] are also capable of handling occlusions, although they are not intended for this purpose.

2.3 Approaches using Deep Learning for Generic Object Detection

Recently, detection approaches based on deep learning have dominated state-of-the-art performance of object detection. One pioneering work based on deep learning for object detection is the framework of regions with convolutional neural network features (R-CNN) [8] in which a bottom-up approach is used to generate a set of sparse region proposals which are classified based on features extracted from a convolutional neural network (CNN). The R-CNN significantly outperforms the previous state-of-the-art approaches based on the DPM [47, 52, 57, 63]. Because of the significant performance improvement achieved by the R-CNN, many approaches have been proposed to improve it by increasing discriminative power of deep neural networks [79–82], generating better proposals [83–86], improving localization accuracy [87–93] and accelerating detection speed [14, 15, 94–97]. The R-CNN and its variants mentioned above adopt a two-stage detection pipeline: region proposal generation followed by classification. YOLO [38], SSD [98], YOLOv2 [95] and RetinaNet [99] represent another popular detection framework based on deep learning. They adopt a one-stage detection pipeline: a CNN is learned to classify dense region proposals at all possible locations on feature maps of the CNN. Compared with two-stage approaches, one-stage approaches are generally faster but less accurate.

2.4 Pedestrian Detection

In the past decade, many efforts have been made to improve pedestrian detection [3, 100–102]. Two major categories of pedestrian detection approaches are channel-feature based approaches [11, 31, 103–108] and deep-learning based approaches [32, 35, 92, 109–116]. For the former, decision trees are usually learned by applying boosting to channel features to form a pedestrian detector. Pedestrian detection is carried out in a sliding-window fashion. For the latter, a deep neural network is trained to either form a pedestrian classifier [32, 35, 92, 109, 110, 112,

115, 116] or generate features which are combined with other types of classifiers for pedestrian detection [111, 113, 114]. This category of approaches usually perform detection by classifying a set of pedestrian proposals. Some recent approaches [115–118] fuse different types of features to improve pedestrian detection performance. Context is also widely exploited to achieve robust pedestrian detection [119–122].

2.5 Occlusion Handling for Pedestrian Detection

Occlusion handling is a difficult problem for pedestrian detection and some approaches have been proposed for this purpose. In [72], a pedestrian template is divided into several blocks and the occlusion is inferred by estimating the visibility statuses of these blocks. An implicit shape model (ISM) [77] is adopted in [123] to generate pedestrian hypotheses in an image with associated supporting regions which are further verified using local and global cues. As each pixel can only be assigned to the supporting region of one hypothesis, this approach is able to separate overlapping pedestrians. Several approaches [24, 31, 44] model occlusions by a set of representative occlusion patterns for each of which a specific detector is learned. The approaches in [24, 44] discover frequently occurring occlusion patterns from annotated training data and train a DPM [10] for each occlusion pattern. In [31], occlusion patterns are manually defined and boosted forests are trained as occlusion-specific detectors. Different from the occlusion patterns defined in [31] which only model a single pedestrian, the occlusion patterns used in [24, 44] can model a single pedestrian or two overlapping pedestrians. For detection, these occlusion-specific detectors are integrated in a winner-take-all fashion. In [45], multi-pedestrian detectors which are learned for detecting overlapping pedestrians are exploited to refine detections obtained by single-pedestrian detectors in a probabilistic framework. Some other approaches [32–36, 109, 124, 125] learn a set of part detectors which are properly integrated for occlusion handling. In [33, 124, 125], a human body is divided into several parts and outputs from part detectors are integrated using a set of rules [124], by linear combination with weights learned from intensity, depth and motion cues [33] or in a Bayesian framework

[125]. Different from the approaches in [33, 124, 125], a human body is represented by a set of overlapping parts which are organized in a hierarchy structure in [34–36, 109]. In [34], a set of rules are defined specifically for the hierarchy structure for inferring occlusion based on part visibility statuses. A discriminative deep model is used to learn correlations among part visibility statuses [35, 109] and the mutual visibility relationship among pedestrians [36] respectively. Then, pedestrian classification is done in a probabilistic framework with detection scores from part detectors as input and part statuses as hidden variables. In [32], a set of part detectors are learned by deep neural networks and integrated by linear combination for occlusion handling. Two approaches based on the DPM [22, 52] are also applicable to occlusion handling for pedestrian detection. They can adaptively switch on or off some parts to handle different occlusion situations. For pedestrian detection, approaches based on the DPM usually perform worse than approaches based on random forests or deep learning.

Chapter 3

Non-rectangular Part Discovery for Generic Object Detection

3.1 Introduction

Although object detection has achieved great success on some specific object categories, *e.g.* human face [7], it is still difficult for existing methods to detect generic object categories that have a wide range of appearance variations. The pictorial structure [126] provides an expressive way to represent objects and has been used in some computer vision applications, such as object recognition [29], pose estimation [127] and action recognition [128]. One of the most successful applications of the pictorial structure is the deformable part-based model (DPM) [10] for object detection. A DPM consists of a root which represents the entire object and several parts which model the appearance of local regions and can move relatively to the root to capture structural deformations. The DPM is a typical part detector ensemble which exploit a set of part detections as well as the whole object region for object detection. To handle viewpoints and subcategories, a set of DPMs are usually learned to form a detector ensemble for object detection [10].

As one of the most popular models for generic object detection, the DPM has received considerable attention from the object detection community and many efforts have been made to

improve it, for example [2, 18, 22, 40, 42, 46, 48, 49, 62]. However, much less work has been done to discover parts for the DPM. Most DPM-based methods adopt the greedy search approach proposed in [10] to initialize a predefined number of parts of rectangular shapes, which may not be optimal for some object categories. Moreover, object structures are not well exploited by the approach. In [40, 129], a three-layer spatial pyramid structure is used to simplify the initialization of parts. An And-Or tree model [49] is proposed to select discriminative part configurations by a dynamic programming algorithm. Although this method can automatically choose proper part sizes, part shapes are still restricted to rectangles.

To address the limitations of the above part discovery approaches for the DPM, we propose a novel data-driven approach to discover non-rectangular parts by exploiting object structures. Following [10], we first learn a root filter for a DPM and then derive parts from the root filter: the root filter is enlarged to twice its original size and a specified number of non-rectangular regions are cropped out of the enlarged root filter as parts such that the obtained part configuration well matches the structure of object examples. We implement our part discovery approach by solving a K -way normalized cuts problem [130] followed by local refinement. Compared with the greedy search approach proposed in [10], our approach has two advantages (See Fig. 3.1 for illustration):

1. The shapes of the parts discovered by our approach can be non-rectangular, which makes the obtained parts more suitable for matching non-rectangular regions;
2. Generally, part configurations obtained by our approach can better fit object structures than those obtained by the greedy search approach.

The effectiveness of our approach is validated on two benchmark datasets, PASCAL VOC2007 [1] and VOC2010 [131].

The remainder of this chapter is organized as follows. The proposed non-rectangular part discovery approach is presented in 3.2. The experimental results are given in 3.3. Section 3.4 concludes the whole chapter.

FIGURE 3.1. Rectangular parts vs non-rectangular parts. (a) Part configuration discovered by the greedy search approach. (c,e,g) Detection examples of the part configuration (a). Red bounding boxes denotes the matching regions of the root filter. (b) Part configuration discovered by our approach. (d,f,h) Detection examples of the part configuration (f). It can be seen that the actual shapes of bicycle seat and handle are better fitted by non-rectangular parts and the part configuration (b) better matches the structure of a left/right facing bicycle.

3.2 Non-rectangular Part Discovery

3.2.1 Deformable Part-based Model

A deformable part model (DPM) [10] is represented by a root filter and some part filters, used for matching the whole region of an object and capturing the appearance of local regions on the object, respectively. The DPM can address appearance variations resulting from deformations and occlusions to some extent. The part filters are allowed to move relatively to the root filter to handle structural deformations. When an object is occluded, some parts can still be detected. Usually, a model of several components each of which is a DPM is learned to handle appearance variations caused by subcategories and viewpoints. In [10], object examples are clustered into several groups according to their viewpoints. As viewpoint annotations are often not available, the object examples are clustered heuristically according to their aspect ratios by k-means. Then, a DPM is learned for each group of object examples and constitutes a component in the model. M_c be the c -th component of the model with N_c part filters. The component M_c is defined by a $(2N_c + 2)$ -tuple $\beta_c = (\mathbf{F}_0, \mathbf{F}_1, \dots, \mathbf{F}_{N_c}, \mathbf{d}_1, \dots, \mathbf{d}_{N_c}, b)$, where $\mathbf{F}_0$ is the root filter, $\mathbf{d}_i \in \mathbb{R}^4$ is the deformation parameters of the part filter $\mathbf{F}_i$ for $1 \leq i \leq N_c$, and b is a bias term. Each filter $\mathbf{F}_i$ for $0 \leq i \leq N_c$ is an $H_i \times W_i$ array of n -dimensional weight vectors, where H_i and W_i are the height and width of $\mathbf{F}_i$, respectively. The 2-tuple $(\mathbf{F}_i, \mathbf{d}_i)$ for $1 \leq i \leq N_c$ forms the i -th part detector in the model. Denote by $P_c(X) = \{\mathbf{p}_0, \mathbf{p}_1, \dots, \mathbf{p}_{N_c}\}$ a matching configuration of the the component M_c on an object example X , where $\mathbf{p}_i = (x_i, y_i)$ is the matching location of $\mathbf{F}_i$ on X . The matching score of the configuration $P_c(X)$ is defined by

$$S(\beta_c, P_c(X)) = \sum_{i=0}^{N_c} \mathbf{F}_i \cdot \phi(X, \mathbf{p}_i) - \sum_{i=1}^{N_c} \mathbf{d}_i \cdot \psi(dx_i, dy_i) + b, \quad (3.1)$$

where $\phi(X, \mathbf{p}_i)$ denotes the histogram of oriented gradients (HOG) features [12] extracted from the matching region of $\mathbf{F}_i$ at the location $\mathbf{p}_i$ on X and $\psi(dx_i, dy_i) = (dx_i^2, dx_i, dy_i^2, dy_i)$ denotes the deformation features with (dx_i, dy_i) the displacement of $\mathbf{F}_i$ relative to its anchor position (i.e. the top-left corner of the part filter) on the root filter. Assume the model has M components

and denote the parameters of these components by $B = (\beta_1, \dots, \beta_M)$. The matching score of the model on the object example X is defined by

$$H(B, X) = \max_{1 \leq c \leq M} \max_{P_c(X) \in Z_c(X)} S(\beta_c, P_c(X)), \quad (3.2)$$

where $Z_c(X)$ is a set of possible matching configurations of the component M_c on the object example X . In other words, the object example X is scored by the most matching component in the model.

Denote by $D = \{(X_i, l_i) | 1 \leq i \leq N\}$ a set of training examples, where X_i is the image region of the i -th example and $l_i \in \{1, -1\}$ indicates whether the example X_i belongs to a specific object category. The model parameters B are learned by minimizing the following objective function

$$L(B, D) = \frac{1}{2} \|B\|^2 + C \sum_{i=1}^N \max(0, 1 - l_i H(B, X_i)). \quad (3.3)$$

As part annotations of the positive training examples are not available, the model cannot be trained directly. Instead, the model is trained in a bootstrapping way. At the beginning, the root filter of each component is obtained by learning a linear support vector machine (SVM) on the positive training examples belonging to the component and some randomly collected negative examples using the HOG features. Then, a set of part filters are obtained by cropping regions from the root filter as described in 3.2.2. The deformation parameters and bias term of each part filter is initialized to $(0.01, 0, 0.01, 0)$ and 0 respectively. So far, each component of the model is initialized. The example clustering results obtained by k-means mentioned above are usually not satisfying enough. Next, the model is used to re-assign positive training examples to different components: each positive training example is assigned to its most matching component according to Eq. (3.1). In the latest version of DPM [132], positive training examples of each component are also flipped horizontally to distinguish left-facing and right-facing poses which cannot be discriminated according to aspect ratios: one original positive training example and its corresponding flipped counterpart are further assigned to two sub-groups, one for left-facing pose and the other for right-facing pose. To do this, one original positive example and its flipped

counterpart are scored by the component and the one with a higher score is assigned to subgroup 1 and the other is assigned to subgroup 2. In this way, the number of components is doubled by flipping the original components in the model (See [132] for details). The model is also used to collect some hard negative examples which have high matching scores. With the new assignment of the positive examples and the collected hard negative examples, the parameters of the model are updated by minimizing Eq. (3.3) on these positive and negative examples. To obtain a good model, the parameters of the model need to be updated several times in each of which the positive examples are re-assigned to the components in the model and a new set of hard negative examples are collected using the current model parameters to re-train the model. We will discuss how to initialize part filters in Sections 3.2.2 and 3.2.3 and refer readers to [10] for more details on model training.

3.2.2 Part Initialization by Greedy Search

The objective function in Eq. (3.3) is non-convex and, as reported in [10], the training process is susceptible to local minima, so it is necessary to select a good initialization of model parameters among which part filters play a very important role. In [10], the part filters of a component are initialized from its root filter in a greedy way: the root filter is enlarged to twice its original size by interpolation and several regions of predefined rectangular shapes (*e.g.* 6×6 squares) that have highest energy are cropped out from the enlarged root filter. The energy of a region in the enlarged root filter is defined by the norm of positive weights in the region. Recall that by definition each location in a root/part filter contains an n -dimensional weight vector. Note that for part selection, negative weights in the enlarged root filter is ignored. Once a region is cropped out as a part filter, the weights in the region are set to zero and next highest-energy region is selected until a specified number of part filters are obtained. This part filter initialization method has two limitations as illustrated in Fig. 3.2: first, the pre-defined rectangular shapes of part filters may not be optimal for some specific object categories, *e.g.* bicycle; second, the part filters obtained in this way may not well fit object structures.

FIGURE 3.2. Toy example of part filter initialization. (a) An object example consisting of two parts. (b) A deformed object example whose two parts move away from each other. (c) The enlarged root filter with two 3×3 highest-energy part filters obtained by the greedy method. Bright cells indicate high-energy regions and dark cells indicate low-energy regions. (d) A better part-filter configuration which agrees with the structure of the two object examples.

3.2.3 Our Approach

To obtain a better initialization of part filters, we propose a data-driven approach to discover part filters which can be non-rectangular by exploiting the structure of object examples.

3.2.3.1 Formulation

Let D_c be the set of object examples belonging to the c -th component and $K = |D_c|$ be the number of object examples in D_c . For each component M_c , we aim to find N_c part filters that have good matching regions on object examples in D_c and are consistent with these examples

in terms of object structure. First, we double the size of the root filter $\mathbf{F}_0$ by interpolation, as in [10], to capture finer details. The enlarged root filter, denoted by $\mathbf{F}'_0$, is represented by a $2H_0 \times 2W_0$ array of cells C_k for $1 \leq k \leq 2H_0 \times 2W_0$, where each cell C_k contains an n -dimensional weight vector in $\mathbf{F}'_0$. Then, from $\mathbf{F}'_0$, we obtain a configuration of N_c connected part filters, $\Lambda = \{\mathbf{F}_i | 1 \leq i \leq N_c\}$, which satisfies the following overlapping constraint:

$$O(\mathbf{F}_i, \mathbf{F}_j) = \frac{\text{Area}(\mathbf{F}_i \cap \mathbf{F}_j)}{\text{Area}(\mathbf{F}_i \cup \mathbf{F}_j)} < \tau \quad \text{for } i \neq j, \quad (3.4)$$

where τ is an overlapping threshold. This constraint prevents any two part filters from overlapping largely. We measure the fitness of the part filter configuration Λ to object examples in D_c by

$$F(\Lambda) = S_R(\Lambda)^\lambda \times S_C(\Lambda), \quad (3.5)$$

where $S_R(\Lambda)$ is the average matching score of Λ over object examples in D_c , $S_C(\Lambda)$ reflects the structural consistency of Λ with these examples, and λ is a parameter used to balance $S_R(\Lambda)$ and $S_C(\Lambda)$. Our goal is to find an optimal part-filter configuration Λ that maximizes $F(\Lambda)$.

Now we describe how to evaluate $S_R(\Lambda)$. We first compute the matching score of each part filter $\mathbf{F}_i$ on a single object example X_j^+ , $R_F(\mathbf{F}_i, X_j^+)$. The root filter $\mathbf{F}_0$ is applied to obtain its optimal matching region R_j on X_j^+ and the region R_j is enlarged to twice its original size. We extract HOG features, denoted by $\mathbf{f}_j$, from the enlarged region R'_j . Due to structural deformation and variation, R'_j may not be able to cover the whole region of the object example. In practice, we extend R'_j outward by a band of 2 cells width, so the final size of $\mathbf{f}_j$ is $(2H_0 + 4) \times (2W_0 + 4)$. The matching score $R_F(\mathbf{F}_i, X_j^+)$ can be computed by

$$R_F(\mathbf{F}_i, X_j^+) = \mathbf{F}_i \cdot \mathbf{f}_j(\mathbf{a}_i + \Delta_i, \mathbf{F}_i), \quad (3.6)$$

where $\mathbf{a}_i$ is the anchor position of $\mathbf{F}_i$ (i.e. the top-left coordinate of $\mathbf{F}_i$ in $\mathbf{F}'_0$), Δ_i is the displacement of $\mathbf{F}_i$ to its optimal matching region in $\mathbf{f}_j$ and $\mathbf{f}_j(\mathbf{a}_i + \Delta_i, \mathbf{F}_i)$ denotes the HOG features extracted from the matching region and has the same shape as $\mathbf{F}_i$. We search for the optimal matching region of $\mathbf{F}_i$ in a neighborhood centered at $\mathbf{a}_i$ in $\mathbf{f}_j$ and do not take deformation

penalty into account. With $R_F(\mathbf{F}_i, X_j^+)$, we define

$$S_R(\Lambda) = \frac{1}{K} \sum_{X_j^+ \in D_c} \sum_{\mathbf{F}_i \in \Lambda} R_F(\mathbf{F}_i, X_j^+). \quad (3.7)$$

Therefore, if part filters in Λ have good matching regions on object examples in D_c , the average score $S_R(\Lambda)$ should be high.

To evaluate $S_C(\Lambda)$, we consider the deformations between cells in $\mathbf{F}'_0$ on object examples in D_c . Let δ_k^j be the displacement of the cell C_k to its optimal matching region on $X_j^+ \in D_c$ for $1 \leq j \leq K$. The optimal matching region is obtained in a neighborhood centered at $(x_k + 2, y_k + 2)$ in $\mathbf{f}_j$, where (x_k, y_k) is the coordinate of C_k in $\mathbf{F}'_0$. To make the estimation of δ_k^j more robust, we search for the optimal matching region of the 3×3 block centered at C_k and take the corresponding displacement as δ_k^j . The deformation between cells C_k and C_l on X_j^+ is measured by

$$d_{kl}^j = \| \delta_k^j - \delta_l^j \|. \quad (3.8)$$

Letting

$$\bar{d}_{kl} = \frac{\sum_{j=1}^K d_{kl}^j}{K} \quad (3.9)$$

be the average deformation between C_k and C_l over all object examples in D_c , we define the cohesiveness between cells C_k and C_l by

$$T(C_k, C_l) = \begin{cases} \exp(-\frac{\bar{d}_{kl}^2}{\sigma^2}) & \text{if } C_k \text{ is adjacent to } C_l; \\ 0 & \text{otherwise,} \end{cases} \quad (3.10)$$

where σ is a parameter and only the cohesiveness between adjacent cells is considered. The total cohesiveness between cells within $\mathbf{F}_i$ and the total cohesiveness between cells in $\mathbf{F}_i$ and cells in $\bar{\mathbf{F}}_i = \mathbf{F}'_0 \setminus \mathbf{F}_i$ are given by

$$T_W(\mathbf{F}_i, \mathbf{F}_i) = \sum_{C_k, C_l \in \mathbf{F}_i} T(C_k, C_l), \quad (3.11)$$

and

$$T_B(\mathbf{F}_i, \bar{\mathbf{F}}_i) = \sum_{C_k \in \mathbf{F}_i, C_l \in \bar{\mathbf{F}}_i} T(C_k, C_l), \quad (3.12)$$

respectively. A cohesive part filter should have strong cohesiveness between its cells (i.e. large $T_W(\mathbf{F}_i, \mathbf{F}_i)$) and relatively weak cohesiveness between its cells and the surrounding cells (i.e. small $T_B(\mathbf{F}_i, \bar{\mathbf{F}}_i)$). Thus, we define the structural consistency of a part filter $\mathbf{F}_i$ by

$$U(\mathbf{F}_i) = \frac{T_W(\mathbf{F}_i, \mathbf{F}_i)}{T_W(\mathbf{F}_i, \mathbf{F}_i) + T_B(\mathbf{F}_i, \bar{\mathbf{F}}_i)}. \quad (3.13)$$

With Eq. (3.13), the structural consistency of the part filter configuration Λ is computed by

$$S_C(\Lambda) = \sum_{\mathbf{F}_i \in \Lambda} U(\mathbf{F}_i). \quad (3.14)$$

3.2.3.2 Optimization

Since there are $(2H_0 \times 2W_0)^{N_c}$ possible part filter configurations, it is computationally expensive to obtain the optimal part-filter configuration Λ^* by exhaustive search. Thus, we solve the optimization problem by first choosing an initial part-filter configuration Λ_0 which has a high structural consistency $S_C(\Lambda_0)$ and then refining Λ_0 to find a local optimum for the objective function in Eq. (3.5). We obtain Λ_0 by partitioning the enlarged root filter $\mathbf{F}'_0$ into N_c part filters such that their total structural consistency is maximized:

$$\Lambda_0 = \arg \max_{\Lambda} S_C(\Lambda) = \arg \max_{\Lambda} \sum_{i=1}^{N_c} \frac{T_W(\mathbf{F}_i, \mathbf{F}_i)}{T_W(\mathbf{F}_i, \mathbf{F}_i) + T_B(\mathbf{F}_i, \bar{\mathbf{F}}_i)}. \quad (3.15)$$

With

$$\frac{T_W(\mathbf{F}_i, \mathbf{F}_i)}{T_W(\mathbf{F}_i, \mathbf{F}_i) + T_B(\mathbf{F}_i, \bar{\mathbf{F}}_i)} = 1 - \frac{T_B(\mathbf{F}_i, \bar{\mathbf{F}}_i)}{T_W(\mathbf{F}_i, \mathbf{F}_i) + T_B(\mathbf{F}_i, \bar{\mathbf{F}}_i)}, \quad (3.16)$$

Eq. (3.15) is equivalent to

$$\Lambda_0 = \arg \min_{\Lambda} \sum_{i=1}^{N_c} \frac{T_B(\mathbf{F}_i, \bar{\mathbf{F}}_i)}{T_W(\mathbf{F}_i, \mathbf{F}_i) + T_B(\mathbf{F}_i, \bar{\mathbf{F}}_i)}. \quad (3.17)$$

FIGURE 3.3. Our part discovery method. (a) The root filter of right-facing bicycles. (b) The average score map of cells. (c) The initial part-filter configuration Λ_0 . The black grids denote the cells which are removed from the enlarged root filter. (d) The final part-filter configuration after refinement. For clarity, $\tau = 0$ is used in this example to obtain the part-filter configuration without overlap.

The objective function in Eq. (3.17) has the same form as the K -way normalized cuts criterion and an approximate solution can be obtained by the multiclass spectral clustering method [130].

Generally, object examples are not exactly rectangular and some cells in the enlarged root filter F'_0 may correspond to the background. These background cells possibly cause small isolated part filters corresponding to the background to be included in the initial part filter configuration Λ_0 . In our method, before partitioning F'_0 , we remove 10% of the cells near the boundary of F'_0 with low average responses over object examples in D_c . The average score of a cell C_k is computed by

$$R_C(C_k) = \frac{\sum_{j=1}^K \mathbf{F}'_0(C_k) \cdot \mathbf{f}_j(C_k)}{K}, \quad (3.18)$$

where $\mathbf{F}'_0(C_k)$ is the weight vector of C_k and $\mathbf{f}_j(C_k)$ is the corresponding HOG feature. In the refinement step, we iteratively update the current part-filter configuration. Possible update

operations include transferring one cell from a part filter to another, removing one cell from a part filter and adding one cell to a part filter. We only choose update operations that satisfy the connectivity constraint on every part filter and the overlapping constraint on the part filter configuration. If the new part filter configuration obtained by an update operation has a higher fitness value defined by Eq. (3.5), we replace the current part-filter configuration with the new one. The refinement process continues until there is no update operation that can lead to a better part-filter configuration. Figure 3.3 illustrates the process of our part discovery method. Figure 3.4 shows the model of the bicycle category which consists of 6 components learned by our approach¹.

3.3 Experiments

We test our approach on two benchmark datasets, PASCAL VOC2007 [1] and VOC2010 [131] datasets, which are commonly used by most recent works for generic object detection. Following the protocol of comp3 [1], we use the *train* and *val* subsets for training and the *test* subset for testing. Average precision (AP) is used as evaluation measure for each object category and mean AP is computed over all object categories. We compare our approach with two most related approaches, DPM-V5 [132] and And-Or [49], which adopt the greedy search approach and the And-Or tree model respectively to initialize part filters for the DPM.

3.3.1 Implementation Details

In our experiments, $\lambda = 0.8$ and $\sigma = 0.5$ are used for all object categories. For the overlap threshold τ , we do not set it to a fixed value. Instead, we restrict each part filter to shrink or expand within a 2-cell width band along the part filter boundary in the refinement step. We find this dynamic setting of τ works well in practice. To make a fair comparison, we use the

¹We cluster positive examples of the bicycle category into 3 groups each of which is further divided into two subgroups as in [132], resulting a model of 6 components. For each group, the components of its two subgroups are mirrors of each other.

FIGURE 3.4. Bicycle model of 6 components. For each component, the root filter is shown to the left of the part filters which are displayed by color polygons.

TABLE 3.1. Average precision (AP) for 10 object categories in the PASCAL VOC2007 dataset with different λ .

	aero	bike	cat	horse	sheep	sofa	bus	dog	bottle	cow
$\lambda = 0.6$	33.9	59.7	20.5	58.8	24.5	38.5	52.9	12.5	27.4	27.3
$\lambda = 0.7$	33.1	60.1	22.3	59.6	22.6	38.4	51.0	12.3	27.4	24.9
$\lambda = 0.8$	34.0	60.2	23.6	59.0	24.2	37.1	53.3	12.8	27.4	26.8
$\lambda = 0.9$	34.5	58.9	22.8	57.0	22.1	38.7	53.6	12.6	27.1	26.3
$\lambda = 1.0$	34.9	59.7	22.5	57.9	23.2	38.9	51.9	12.6	26.8	25.2

same setting as DPM-V5 for part initialization: the number of part filters for each component is set to 8 and the deformation parameters of each part filter are set to $[0.1, 0, 0.1, 0]$. After its parameters are initialized, the model is trained in the same way as in DPM-V5. Although the shapes of parts in our model are not rectangular, for simplicity we still represent each part by a rectangular filter with a corresponding mask to disable the unused cells. As our model has roughly the same number of cells in all part filters as the model obtained by DPM-V5, the computational complexity of our method is similar to that of DPM-V5.

3.3.2 Parameter Selection

We test our part discovery approach on 10 object categories selected from PASCAL VOC2007 with λ ranging from 0.6 to 1.0. The results are listed in Table 3.1. It can be seen that a bad choice of λ may lead to a significant performance decrease for some object categories. Take the cat category for example. The performance of $\lambda = 0.6$ decreases by 13% compared to that of $\lambda = 0.8$. As shown in Table 3.1, $\lambda = 0.8$ works well for most of the 10 object categories. We thus use this setting of λ to test our approach on the PASCAL VOC2007 and VOC2010 datasets. Cross-validation can also be used to select an appropriate λ for each object category.

3.3.3 Results on 20 Object Categories

Detection results of DPM-V5, And-Or and our approach on the PASCAL VOC2007 and VOC2010 datasets are given in Tables 3.2 and 3.3 respectively. All the results are obtained without any

TABLE 3.2. Performance comparison using Average Precision (AP) on 20 object categories in the PASCAL VOC2007 dataset.

	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table
DPM-V5	32.4	57.7	10.7	15.7	25.3	51.3	54.2	17.9	21.0	24.0	25.7
And-Or	35.3	60.2	11.0	16.6	29.5	53.0	57.1	23.0	22.9	27.7	28.6
Ours	34.0	60.2	12.2	18.3	27.4	53.3	55.9	23.6	22.6	26.8	30.8
FRCN+ResNet [81]	79.8	80.7	76.2	68.3	55.9	85.1	85.3	89.8	56.7	87.8	69.4
	dog	horse	motor	person	plant	sheep	sofa	train	tv	mean	
DPM-V5	11.6	55.6	47.5	43.5	14.5	22.6	34.2	44.2	41.3	32.5	
And-Or	13.1	58.9	49.9	41.4	16.0	22.4	37.2	48.5	42.4	34.7	
Ours	12.8	59.0	49.5	42.6	15.4	24.2	37.1	44.8	43.4	34.7	
FRCN+ResNet [81]	88.3	88.9	80.9	78.4	41.7	78.6	79.8	85.3	72.0	76.4	

post-processing, like bounding-box prediction or context rescoring. Our approach outperforms DPM-V5 for 19 and 17 out of 20 classes in PASCAL VOC2007 and VOC2010 respectively, which demonstrates the advantage of the use of non-rectangular part filters. Our approach achieves significant performance improvements for a few object categories, like bicycle, dining table and tv monitor, which have relatively stable structures. Overall, our approach is comparable to And-Or and performs slightly better than And-Or on PASCAL VOC2010. Besides the use of non-rectangular part filters, another difference between our approach and And-Or is that our approach uses 8 part filters for all object categories, while And-Or can automatically determines the number of part filters for each object category. We will study part number selection for the DPM in future work to check if it can further improve the performance. Figure 3.5 shows some detection examples of our approach for the 20 object categories in PASCAL VOC2007. We also add the results of a typical deep learning based approach, Faster R-CNN with ResNet (FRCN+ResNet) [81], to Table 3.2 for reference.

3.4 Conclusion

In this chapter, we present a data-driven approach to discover parts for the classic part detector ensemble DPM. Different from most DPM-based methods which use a specified number of rectangular parts, our approach is capable of discovering non-rectangular parts which can

FIGURE 3.5. Detection examples of our approach for the 20 object categories in the PASCAL VOC2007 dataset.

TABLE 3.3. Performance comparison using Average Precision (AP) on 20 object categories in the PASCAL VOC2010 dataset.

	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table
DPM-V5	42.9	47.2	10.3	11.1	26.3	48.4	40.2	22.9	17.0	22.9	10.2
And-Or	44.6	48.5	10.8	12.9	26.3	47.5	41.6	21.6	17.3	23.6	11.5
Ours	45.9	50.7	10.4	11.2	28.7	50.5	44.2	24.2	17.4	24.0	13.7
	dog	horse	motor	person	plant	sheep	sofa	train	tv	mean	
DPM-V5	19.9	41.5	44.0	41.0	7.6	28.3	18.2	39.0	32.9	28.6	
And-Or	22.9	40.9	45.3	37.9	9.6	30.4	25.3	39.0	31.2	29.4	
Ours	17.6	40.2	45.7	38.8	8.3	29.3	20.2	41.0	35.6	30.0	

better fit object structures. Our approach can be efficiently implemented by solving a K -way normalized cuts problem followed by local refinement. The effectiveness of the proposed approach is validated on the PASCAL VOC2007 and VOC2010 datasets. In future work, we will explore how to automatically select the part number for each object category.

Chapter 4

Occlusion Pattern Discovery for Generic Object Detection

4.1 Introduction

Partial occlusions occur frequently and present a great challenge for object detection in practical applications. As pointed out in [3, 133], most existing object detection methods have difficulty detecting objects that are partially occluded. For detectors like [10, 12] which perform object detection by template matching, occlusions may introduce negative responses at occluded regions and therefore decrease the overall response of these detectors on a partially occluded object, which possibly causes the occluded object to be missed. Learning a detector ensemble [2, 23, 31] whose components are responsible for handling different occlusion situations can reduce the effects of occlusions, therefore improving detection performance. In this chapter, we explore how to learn a detector ensemble based on a classic deformable part model (DPM) [10], to handle occlusions for object detection.

Mixture models [2, 22–24, 31, 33, 44, 45, 134] have been widely used for handling occlusions. One key step of building a detector ensemble is to obtain a set of occlusion patterns¹ each of

¹An occlusion pattern is a pattern that describes which portion of an object is visible and which portion is occluded.

FIGURE 4.1. Overview of the proposed approach to occlusion pattern discovery applied to the person category.

which constitutes a component in the detector ensemble. These occlusion patterns are of great importance to the quality of the detector ensemble. In [2, 31, 33, 34], occlusion patterns are manually designed according to prior knowledge. For example, occlusion patterns of humans are designed to gradually increase the amount of occlusion from the bottom [2]. This way of designing occlusion patterns may not be optimal and is not easy to extend to general object categories which usually have different occlusion patterns that are scenario dependent. Thus, it is preferred to automatically discover occlusion patterns that can capture different occlusion situations in training samples. Clustering is another way to obtain occlusion patterns, e.g. based on bounding-box annotations [24, 44, 45], 3D object data [23, 74], or part-level annotations [22, 52]. The occlusion patterns obtained by clustering are generally representative among object examples, but the discriminative power of individual occlusion patterns is not taken into account. Non-discriminative occlusion patterns would possibly bring many false positives at the detection stage, thus are not preferred. In addition, some approaches use additional data like part annotations [22, 52] and 3D object data [23, 74] which are usually difficult to obtain, therefore restricting their scalability to general object categories.

To address the above issues, we propose a data-driven approach which applies to general object categories and is capable of discovering occlusion patterns that are both representative and discriminative. Different from previous works which require part annotations [22, 23, 52], our

approach only needs bounding-box annotations which are relatively easy to obtain. Figure 4.1 shows the overview of the proposed approach applied to the person category. First, we train a full model which is a DPM [10] with fully visible person examples. Then, we generate a pool of occlusion patterns from the full model according to part visibility states. In our approach, an occlusion pattern consists of a subset of visible parts. As each part can be visible or occluded, m parts would generate a total number of 2^m patterns. Next, we select from the large pattern pool a subset of occlusion patterns which can well represent positive examples and distinguish these examples from negative ones. We model the subset selection as a facility location problem. Thanks to its submodular property, greedy search [135] can be applied to obtain a near-optimal solution and is known to work well in practice. Our approach can be easily extended to other object categories as described in Section 4.2.3.

The occlusion patterns discovered by our approach are themselves DPMs and can be used for object detection and occlusion reasoning when they are properly tuned. We adopt the weak-label structural SVM (WL-SSVM) framework [2] to update the model parameters of the discovered occlusion patterns. Given an image, the tuned occlusion patterns can be applied in a sliding-window fashion as in [10] to collect detections each of which is matched to one occlusion pattern. The part-level occlusion of a detection can be estimated according to its matching occlusion pattern.

Since the detection framework of regions with convolutional neural networks (R-CNN) [8] was proposed, many efforts have been made to increase discriminative power of deep neural networks [80–82, 121], generate better proposals [83–86], improve localization accuracy [88–93, 108] and accelerate detection speed [14, 15, 38, 94–98]. Given an image, these approaches can only detect regions in the image which contain objects of interest. They are less concerned about whether objects are occluded and how they are occluded, which could be important for applications like robotics. Although the discovered occlusion patterns by our approach are based on the hand-crafted histogram of oriented gradients (HOG) [12], they can be combined with state-of-the-art detectors such as Faster R-CNN [15], SSD [98], YOLO [38, 95] for improving detection performance and achieving part-level occlusion reasoning. Take Faster

FIGURE 4.2. Occlusion reasoning. The red bounding box in (a) is a candidate detection which is obtained by Faster R-CNN. The candidate detection is matched to the occlusion pattern shown in (b). Green and blue bounding boxes in (a) show visible parts and the estimated full human body respectively after the candidate detection is aligned with the occlusion pattern.

R-CNN for example. For a specific object category, we can apply Faster R-CNN to collect a set of candidate detections in an image which are then matched to the discovered occlusion patterns for occlusion reasoning. Specifically, for a partially occluded object, we can estimate the region of its full body as well as its visible parts (See Fig. 4.2 for illustration). Based on the information from the two types of detectors, Faster R-CNN and our occlusion patterns, we learn a non-linear SVM to rescore these candidate detections, which can further improve the performance compared with Faster R-CNN.

We validate the effectiveness of the discovered occlusion patterns on PASCAL VOC2007 [1] and VOC2010 [131] datasets. In summary, our contributions are two-fold: (1) we propose a novel approach to discover representative and discriminative occlusion patterns for general object categories only using bounding-box annotations; (2) we demonstrate that the occlusion patterns

discovered by our approach can be applied independently or combined with state-of-the-art detectors for improving detection performance and achieving part-level occlusion reasoning. The remainder of this chapter is organized as follows. Section 4.2 presents the proposed occlusion pattern discovery approach. Section 4.3 describes how the discovered occlusion patterns are used for object detection and part-level occlusion reasoning. The experimental results are given in Section 4.4. Section 4.5 concludes the whole chapter.

4.2 Occlusion Pattern Discovery

Our approach to occlusion pattern discovery first generates a pool of candidate occlusion patterns and then selects from the pool a subset of occlusion patterns that are both representative and discriminative (See Fig. 4.1 for the overview of the proposed approach). In Sections 4.2.1 and 4.2.2, we describe how occlusion patterns are discovered for one viewpoint of an object category by taking the person category for example. Then, we extend the approach to multiple viewpoints in Section 4.2.3.

4.2.1 Generating candidate occlusion patterns

As in [2, 22, 23], we model occlusion at part level. The difference is that we automatically obtain parts with object-level annotations instead of manually defining parts [2] and using part-level annotations [22, 23] which are usually difficult to label. This makes it easy to apply our approach to general object categories.

First, we train a deformable part model (DPM) [10] with m parts on a set of fully visible person examples without distinguishing different viewpoints. We consider a person example that has a standing or near-standing pose and is not occluded by any stuff to be fully visible. We will describe how to obtain fully visible person examples in Section 4.4.1. Figures 4.1(a) and 4.1(b) show some fully visible person examples and the full model respectively. Denote the full model by $M^{full} = (f_0, \dots, f_m, a_1, \dots, a_m, d_1, \dots, d_m, b)$, where f_0 is the root filter, f_i for

FIGURE 4.3. Example of occlusion pattern. (a) The occlusion pattern has two visible parts marked by blue and green rectangles respectively. (b) All weights in the invisible portion of the root filter are equal to 0. (c) Weights in the visible portion (the top area) are set to 0 and cells close to the boundary of the visible region have large weights. (d) The occluder filter of the occlusion pattern has zero weights in the visible portion.

$1 \leq i \leq m$ is a part filter, a_i and d_i are the anchor position and deformation parameters of the part filter f_i respectively, and b is a bias term. By defining $F^{full} = \{f_0, \dots, f_m\}$, $A^{full} = \{a_1, \dots, a_m\}$ and $D^{full} = \{d_1, \dots, d_m\}$, we can represent the full model by a 4-tuple $M^{full} = (F^{full}, A^{full}, D^{full}, b)$.

Next, we generate a pool of occlusion patterns from the full model M^{full} by enumerating all possible occlusion patterns. For each part i , we define a visibility state $v_i \in \{0, 1\}$ which indicates if the i -th part is visible. A vector $V = (v_1, \dots, v_m)$ specifies an occlusion pattern which has one root filter and $\sum_{i=1}^m v_i$ visible parts. The other invisible parts are discarded. We require that an occlusion pattern should have at least one visible part, so there are in total $2^m - 1$ possible occlusion patterns. All the occlusion patterns share the root filter f_0 and part filters f_i ($1 \leq i \leq m$) from the full model M^{full} , but they have their own deformation parameters and bias terms. We also associate each occlusion pattern P_j ($1 \leq j \leq 2^m - 1$)

with a mask R_j^r which has the same dimensions as the root filter and specifies the visible portion of P_j in the root filter. By placing the parts of P_j at their anchor positions on the root filter, we define the region in the root filter covered by the parts as the visible portion. The weights in the visible portion of R_j^r are set to 1 and the remaining weights are set to 0. With the mask R_j^r , the root filter of P_j is constructed by $f_0 \otimes R_j^r$, where $\otimes$ donates element-wise multiplication. Figure 4.3(a,b) show an example of occlusion pattern. Let $V_j = \{v_{j1}, \dots, v_{jm}\}$ be the vector specifying the occlusion pattern P_j and $m_j = \sum_{i=1}^m v_{ji}$ be the number of parts in P_j . We can define P_j by a 5-tuple $P_j = (F_j, A_j, D_j, b_j, R_j^r)$, where $F_j = \{f_0\} \cup \{f_i | v_{ji} = 1\}$, $A_j = \{a_i | v_{ji} = 1\}$ and $D_j = \{d_{jk} | 1 \leq k \leq m_j\}$ with d_{jk} the deformation parameters of the k -th part filter of P_j . For convenience, we denote the k -th part filter of P_j by f_{jk} . According to the construction, each occlusion pattern is a DPM. The generated pool of occlusion patterns is defined by $C = \{P_j | 1 \leq j \leq 2^m - 1\}$.

4.2.2 Selecting occlusion patterns

Among the generated occlusion patterns, some may rarely occur and some are not discriminative for distinguishing person examples from non-person image patches. Instead of using all the occlusion patterns, we select from the pattern pool C a subset G of occlusion patterns which are both representative and discriminative. Let $X = \{x_1, \dots, x_n\}$ be a set of person examples, $Y = \{y_1, \dots, y_n\}$ be their associated bounding-box annotations and K be the maximum number of occlusion patterns to be chosen. We model the subset selection as a facility location problem [136, 137] by maximizing the following objective function:

$$\begin{aligned}
 R(G) &= \sum_{i=1}^n \max_{P_j \in G} S(P_j, x_i) - \lambda \sum_{P_j \in G} V(P_j), \\
 \text{s.t. } &G \subseteq C, |G| \leq K,
 \end{aligned} \tag{4.1}$$

where $S(P_j, x_i)$ is the matching score of the occlusion pattern P_j on the person example x_i , $V(P_j)$ is the cost of selecting P_j and λ is a constant. The use of the first term in Eq. (4.1) is to select a set of representative occlusion patterns such that each person example in X well

matches at least one selected occlusion pattern. The cost $V(P_j)$ is defined as $V(P_j) = 1 - T(P_j)$, where $T(P_j)$ denotes the discriminative power of P_j . According to the definition of $V(P_j)$, discriminative occlusion patterns are preferred. We can consider an occlusion pattern P as a candidate site for locating a facility and a person example x as a client located at a specific location. $V(P)$ can be interpreted as the cost of opening a facility at P and $S(P, x)$ is the profit made from the facility by serving the client x . The goal is to maximize the overall profit.

We compute $S(P_j, x_i)$ by finding an optimal placement of P_j on x_i . First, we construct an HOG feature pyramid [10] for x_i denoted by $\Phi(x_i)$. Let $p_j = (p_{j0}, \dots, p_{jm_j})$ be a placement of P_j on $\Phi(x_i)$, where $p_{jk} = (x, y, l)$ specifies the matching location of the k -th filter of P_j on $\Phi(x_i)$. The score of the placement p_j is defined by

$$U(P_j, x_i, p_j) = \sum_{k=0}^{m_j} f_{jk} \cdot \Phi(x_i, p_{jk}) + \sum_{k=1}^{m_j} d_{jk} \cdot \Psi(p_{jk}, p_{j0}) + b_j, \quad (4.2)$$

where $\Phi(x_i, p_{jk})$ and $\Psi(p_{jk}, p_{j0})$ denote the HOG features and the deformation features corresponding to the placement of f_{jk} at p_{jk} respectively (See [10] for details). Denote by $W(p_j)$ the detection window derived from the placement p_j . The matching score $S(P_j, x_i)$ is defined by

$$S(P_j, x_i) = \max_{p_j} U(P_j, x_i, p_j), \quad (4.3)$$

$$\text{s.t. } \text{Overlap}(W(p_j), y_i) = \frac{\text{Area}(W(p_j) \cap y_i)}{\text{Area}(W(p_j) \cup y_i)} \geq \tau, \quad (4.4)$$

where τ is an overlap threshold. If P_j does not have any feasible placement on x_i , $S(P_j, x_i) = \gamma$ where γ is a small constant. Let Q be the set of all feasible matchings (x_i, P_j) . We set γ as follows

$$\gamma = \min_{(x_i, P_j) \in Q} S(P_j, x_i).$$

The score of an infeasible matching is lower than the score of any feasible matching. The constraint in Eq. (4.4) ensures that if an occlusion pattern can match x_i , there should be at least one placement that has sufficient overlap with the ground-truth bounding-box y_i , which can reduce mismatching between occlusion patterns and positive examples. For example, with this

constraint, a full-body pattern cannot match a person example with only head and shoulders visible.

For $T(P_j)$, we choose average precision (AP) as the criterion. The AP of the occlusion pattern P_j is evaluated on the fully visible examples and 400 negative images which do not contain any positive examples. Here, we use the fully visible examples instead of the whole set of positive examples, since we can estimate the ground-truth bounding boxes of each occlusion pattern on the fully visible examples. Given a fully visible example, we roughly estimate the bounding box of P_j according to its visible portion in the root filter f_0 , i.e. the visible portion in f_0 is transferred to the corresponding region in the bounding box of the fully visible example by proper scaling. We evaluate the AP of P_j as follows: first, we compute the matching scores of P_j on fully visible examples using Eq. (4.3) and Eq. (4.4) with the estimated bounding boxes; then, we apply P_j to negative images to collect false positives; finally, the AP is calculated according to the matching scores of true positives and false positives.

Now, we introduce how to maximize the objective function in Eq. (4.1) to obtain the subset of occlusion patterns G . Direct optimization of the objective function is NP-hard and therefore intractable. It is easy to show that the objective function satisfies the following submodularity property

$$R(A \cup \{P\}) - R(A) \geq R(B \cup \{P\}) - R(B), \quad (4.5)$$

for any subsets $A, B \subseteq C$, $A \subset B$, and $P \notin B$. With this property, the optimization problem can be efficiently solved by a greedy algorithm [135] to obtain a near-optimal solution. The algorithm starts with an empty set $G = \emptyset$. At each iteration, the occlusion pattern $P \notin G$ with the maximum marginal gain $R(G \cup A) - R(G)$ is added to G , $G = G \cup \{P\}$. This operation repeats until $|G| = K$ or the maximum marginal gain is negative. This algorithm can obtain an approximate solution with a theoretical lower bound and proves to work well in practice.

Let $\mathcal{D} = (\mathcal{I}, \mathcal{Y})$ be the training data, where $\mathcal{I} = \{I_1, \dots, I_n\}$ is a set of images and $\mathcal{Y} = \{Y_1, \dots, Y_n\}$ is a set of bounding-box annotations. Y_i is a set of bounding boxes specifying the regions of object examples in I_i . $Y_i = \emptyset$ if I_i does not contain any object examples. Algorithm

Algorithm 1: Single-viewpoint occlusion pattern discovery**Input:**Training data $\mathcal{D} = (\mathcal{I}, \mathcal{Y})$;The number of parts, m ;The maximum number of occlusion patterns to be selected, K ;**Output:**A group of occlusion patterns, G ;**1: Stage 1: Generating occlusion patterns**2: Collect fully visible examples X^{full} and negative examples X^- from $\mathcal{D}$;3: Train a full model M^{full} with m parts on X^{full} and X^- ;4: Generate a set of occlusion patterns C from M^{full} ;**5: Stage 2: Selecting occlusion patterns**6: Collect positive examples X^+ from $\mathcal{D}$;7: For each $P_j \in C$ and $x_i^+ \in X^+$, precompute $S(P_j, x_i^+)$ according to Eq. (4.3) and Eq. (4.4);8: For each $P_j \in C$, precompute its discriminative power $V(P_j)$ on $\mathcal{D}$;9: $G \rightarrow \emptyset$;10: **for** $i = 1, \dots, K$ **do**11: $P^* = \arg \max_{P \in C} R(G \cup \{P\}) - R(G)$;12: $g^* = R(G \cup \{P^*\}) - R(G)$;13: **if** $g^* > 0$ **then**14: $G = G \cup \{P^*\}$ 15: **end if**16: **end for**17: **return** G ;

1 summarizes the procedure of single-viewpoint occlusion pattern discovery. Table 4.1 lists the variables used in this Section.

4.2.3 Extending to multiple viewpoints

The proposed approach works well on the person category without distinguishing different viewpoints. However, it is usually not the case for some other object categories like car, since the appearances of these object categories vary largely when viewed from different viewpoints. Our approach can be easily extended to multiple viewpoints. First, we cluster the positive examples of an object category into several viewpoints using either the heuristic approach based on aspect ratio [10] or more sophisticated viewpoint classification methods like [138]. Then, for each viewpoint, we train a full model from which a set of occlusion patterns are generated

Symbol	Description
m	The number of parts in the full model
f_0	The root filter of the full model
f_i	The i -th part filter of the full model
a_i	The anchor position of f_i relative to f_0
d_i	The deformation parameters of f_i
b	The bias term of the full model
M^{full}	The full model $M^{full} = (F^{full}, A^{full}, D^{full}, b)$
F^{full}	The set of filters in M^{full} , $F^{full} = \{f_0, \dots, f_m\}$
A^{full}	The set of anchors in M^{full} , $A^{full} = \{a_1, \dots, a_m\}$
D^{full}	The set of deformation parameters in M^{full} , $D^{full} = \{d_1, \dots, d_m\}$
C	The set of all occlusion patterns
P_j	The j -th occlusion pattern in C , $P_j = (F_j, A_j, D_j, b_j, R_j^r)$
v_{ji}	The visibility state of the i -th part in P_j
V_j	The visibility vector of P_j , $V_j = \{v_{j1}, \dots, v_{jm}\}$
R_j^r	The visible portion of P_j in f_0
m_j	The number of parts in P_j
F_j	The set of filters in P_j
A_j	The set of anchors in P_j
D_j	The set of deformation parameters P_j
b_j	The bias term in P_j
X^+	A set of positive examples
x_i^+	The i -th positive example in X^+
I_i	The image where x_i^+ is obtained
y_i	The bounding box specifying the region of x_i^+ in I_i
G	A group of occlusion patterns selected from C
$R(G)$	The objective value for G indicating how good the occlusion patterns in G are
$S(P_j, x_i^+)$	The matching score of P_j on x_i^+
$T(P_j)$	The discriminative power of P_j
K	The maximum number of occlusion patterns to be chosen
$\Phi(x_i^+)$	The HOG feature pyramid constructed for x_i
p_j	A placement of P_j on $\Phi(x_i^+)$, $p_j = (p_{j0}, \dots, p_{jm_j})$
p_{jk}	The matching location of the k -th part filter of P_j on $\Phi(x_i^+)$
$W(p_j)$	The detection window derived from the placement p_j

TABLE 4.1. Summary of variables for occlusion pattern discovery.

as described in Section 4.2.1. We will describe how to obtain fully visible examples to train the full models in Section 4.4.2. Finally, the occlusion patterns from different viewpoints are merged into a single set and the pattern selection method in Section 4.2.2 is applied to obtain a subset of representative and discriminative occlusion patterns.

4.3 Object detection with discovered occlusion patterns

After obtaining a subset of occlusion patterns, we first train a deformable part model (DPM) [10] for each occlusion pattern² and use these DPMs for object detection and occlusion reasoning.

4.3.1 Modeling occluders

As demonstrated in [2], proper modeling of occluder appearances can improve detection performance. Similar to [2], we use an occluder filter to model occluder appearances. In [2], occlusion patterns share the same occluder filter which is placed at the bottom of each occlusion pattern. Occlusions may occur at other places besides the bottom, therefore simply placing the occluder filter at the bottom is not suitable. Let f^o be the occluder filter which has the same dimensions as the root filter f_0 . We create an occluder mask R_j^o for each occlusion pattern P_j . In the mask R_j^o , the weights corresponding to the visible portion of P_j are set to 0. The other weights are set according to the distance of each cell to the boundary of the visible portion. Here, we call one position in the mask a cell and the distance is measured by Manhattan or city block distance. Let d be the distance of a cell to the boundary of the visible portion. The weight of the cell is set to 2^{-d} . The cells that are close to the boundary of the visible portion are given high weights. With the mask R_j^o , the occluder filter of P_j , denoted by f_j^o , is constructed by $f_j^o = R_j^o \otimes f^o$. An example of occluder mask and filter is given in Fig. 4.3(c,d). The occluder

²An occlusion pattern itself is a DPM (See the definition of occlusion pattern in Section 4.2.1). By training, we update the model parameters of the occlusion pattern.

filter f^o is placed at the same position as the root filter, i.e. $a^o = a_0$, where a^o denotes the anchor position of f^o . Let d_j^o denote the deformation parameters of f_j^o . With occluder modeling³, the occlusion pattern P_j is represented by $P_j = (F'_j, A'_j, D'_j, b_j, R_j^r, R_j^o)$, where $F'_j = F_j \cup f^o$, $A'_j = A_j \cup a^o$ and $D'_j = D_j \cup d_j^o$.

4.3.2 Learning model parameters

We integrate these occlusion patterns into a detector ensemble which is trained using the weak-label structural SVM (WL-SSVM) framework [2]. Denote the detector ensemble by $M^{mix} = \bigcup_{j: P_j \in G} P_j$ and let $\Omega(P_j, x_i)$ be the set of possible placements of the occlusion pattern P_j on the example x_i . The set of possible placements of the detector ensemble on x_i is defined by $\Omega(M^{mix}, x_i) = \bigcup_{P_j \in G} \Omega(P_j, x_i)$. Let w be a vector obtained by concatenating all the parameters of M^{mix} . The model parameters w are learned by minimizing the following objective function:

$$E(w) = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n L(w, x_i, y_i), \quad (4.6)$$

where $L(w, x_i, y_i)$ is the surrogate training loss defined by

$$L(w, x_i, y_i) = \max_{p \in \Omega} [w \cdot \varphi(M^{mix}, x_i, p) + L_{margin}(y_i, p)] - \max_{p \in \Omega} [w \cdot \varphi(M^{mix}, x_i, p) - L_{output}(y_i, p)], \quad (4.7)$$

where $\Omega = \Omega(M^{mix}, x_i)$ and $\varphi(M^{mix}, x_i, p)$ is the feature representation corresponding to the placement p on x_i . During training, L_{margin} pushes down the scores of high-loss placements, while L_{output} pulls up the scores of low-loss placements. Define a loss function $L_{l,\tau}(y, p)$ by

$$L_{l,\tau}(y, p) = \begin{cases} l & \text{if } y = \perp \text{ and } p \neq \perp \\ 0 & \text{if } y = \perp \text{ and } p = \perp \\ l & \text{if } y \neq \perp \text{ and } \text{overlap}(y, W(p)) \leq \tau \\ 0 & \text{if } y \neq \perp \text{ and } \text{overlap}(y, W(p)) > \tau \end{cases} \quad (4.8)$$

³Without the occluder, P_j is represented by $P_j = (F_j, A_j, D_j, b_j, R_j^r)$ in Section 4.2.1

where $\perp$ represents the background output and $W(p)$ denotes the detection window derived from the placement p . We use $L_{margin} = L_{1,0.5}$ and $L_{output} = L_{1,\infty}$. We refer readers to [2] for more details on WL-SSVM.

4.3.3 Object detection with occlusion inferring

We explore two ways of applying the discovered occlusion patterns for object detection.

4.3.3.1 Sliding-window search

The discovered occlusion patterns can be applied in a sliding-window fashion for object detection as in [10]. Given an image, a pyramid of HOG features is constructed and the responses of these occlusion patterns are evaluated at each position in each layer of the HOG feature pyramid. Assume that k occlusion patterns are selected. Each position in a layer generates k candidate regions each of which corresponds to one occlusion pattern. Then, non-maximum suppression is applied to all the candidate regions to discard duplicate detections. The part-level occlusion of a detection, i.e. which parts of the detection are occluded, can be estimated according to the corresponding occlusion pattern of the detection.

4.3.3.2 Combination with another detector

The discovered occlusion patterns can also be combined with another detector to improve the performance and enable occlusion reasoning for the latter. Take Faster R-CNN [15] for example. First, Faster R-CNN is applied to an image to obtain a set of candidate detections for a specific object category. Each candidate detection is aligned with the discovered occlusion patterns respectively using Eq. (4.3) and Eq. (4.4). Let q be a candidate detection and $s(q)$ be the detection score of q given by Faster R-CNN. Denote by j^* the index of the most matching occlusion pattern of q , i.e. $j^* = \arg \max_j S(P_j, q)$ subject to $P_j \in G$. Let $p_{j^*}^*$ be the optimal placement of the occlusion pattern P_{j^*} on q . We can obtain from $p_{j^*}^*$ an estimation of the part-level occlusion of the candidate detection q . We represent q by a vector

$u = (s(q), S(P_{j^*}, q), \text{Overlap}(W(p_{j^*}^*), q))$, where $\text{Overlap}(W(p_{j^*}^*), q)$ defined in Eq. (4.4) reflects the degree of consistency between the occlusion pattern P_{j^*} and the candidate detection q . Then, we train a non-linear SVM with a radial basis function as the kernel to rescore q by

$$s'(q) = \sum_{i=1}^n \alpha_i \kappa(u, u_i), \quad (4.9)$$

where n is the number of training examples, u_i is the representation of the i -th training example and $\kappa(u, u_i)$ is defined by

$$\kappa(u, u_i) = \exp\left(\frac{-\|u - u_i\|^2}{\sigma^2}\right). \quad (4.10)$$

After all the candidate detections are rescored by Eq. (4.9), non-maximum suppression is applied to discard duplicate detections.

4.4 Experiments

4.4.1 Single-viewpoint occlusion pattern discovery

4.4.1.1 Implementation

We test our approach on the person category in the VOC2007 dataset without distinguishing different viewpoints. As the dataset only provides bounding-box annotations specifying the regions of persons in images, we do not know if a person example is occluded or not. We obtain fully visible person examples using the heuristic method in [2]: person examples are clustered into three groups according to aspect ratios of these examples and the group with the maximum average aspect ratio is selected. Then we train a full model of two components which are mirror images of each other using the code of DPM (version 5) [132]. Each component has 1 root filter and 8 part filters. The size of the root filter is 15×6 and the size of each part filter is 8×8 . For occlusion pattern selection, we set $K = 6$ to select exactly 6 occlusion patterns from 255 candidates to make a direct comparison with the grammar model (GRAM) [2]. We use the implementation of WL-SSVM in [132] to train our model with $C = 0.003$.

FIGURE 4.4. Comparison of different approaches using HOG features on the person category in VOC2007.

4.4.1.2 Experimental results

Figure 4.4 shows precision-recall curves of our approach and two baseline approaches, DPM and GRAM. Detection performance is summarized by average precision (AP). It can be seen that both GRAM and our approach outperform DPM, demonstrating that proper occlusion handling can improve detection performance. Our approach outperforms GRAM by 0.4%. The performance improvement is not large, because the occlusion patterns discovered by our approach are similar to those manually designed for GRAM as shown in Fig. 4.5. The first four occlusion patterns of ours have the same property as the 6 occlusion patterns of GRAM: the occlusion increases gradually from bottom to top. Besides these bottom-occluded

patterns, our approach discovers two more common patterns in which the occlusion occurs on either side. To validate the effectiveness of the design aspects of our approach, we compare

FIGURE 4.5. Occlusion patterns of GRAM [2] (Column 1) and our approach (Columns 2 to 7).

FIGURE 4.6. Person detection examples. Columns 1-4 show some examples which are matched with correct occlusion patterns. Columns 5-6 show some failure examples which are matched with wrong occlusion patterns. For clarity, only one detection is displayed in each image.

three different implementations, OCCL, OCCL-D and OCCL-DO. OCCL and OCCL-D are two implementations without occluder modeling. OCCL only uses the first term in Eq. (4.1) for occlusion pattern selection, while OCCL-D uses both terms. By taking discriminative power of occlusion patterns into account, the performance of our approach is improved by 0.7% from 45.6% (OCCL) to 46.3% (OCCL-D). With occluder modeling, another performance improvement of 0.8% is achieved from 46.3% (OCCL-D) to 47.1% (OCCL-DO). For GRAM, the occluder filter improves its performance by 1.2% from 45.5% to 46.7%. Figure 4.6 shows some

person detection examples of our approach.

4.4.2 Multiple-viewpoint occlusion pattern discovery

4.4.2.1 Implementation

For the case of multiple viewpoints, we evaluate our approach on 20 object categories in VOC2007 and VOC2010 datasets. As viewpoint annotations are not available in the datasets, we simply cluster positive examples of an object category into three groups using aspect ratios of these examples as in [10]. Each group corresponds to one viewpoint. This viewpoint clustering method is not precise, especially when some examples are partially occluded. To refine the clustering result, we first train a DPM for each viewpoint with the examples in the corresponding group. Then, we re-assign the positive examples to the three viewpoints with the obtained DPMs: we compute matching scores of each DPM on these examples; each example is assigned to the DPM with the highest matching score. Instead of training these DPMs separately, we train them jointly in a single SVM by minimizing the following max-regularized objective function as in version 5 of DPM [132]:

$$\max_{1 \leq i \leq 3} \|\beta_i\|^2 + C \sum_{i=1}^n \max(0, 1 - y_i g(\beta, x_i)), \quad (4.11)$$

where β_i is model parameters of the i -th DPM, $\beta = (\beta_1, \beta_2, \beta_3)$ and $g(\beta, x_i)$ is the score of the training example x_i . With this objective function, the norms $\|\beta_i\|$ for $1 \leq i \leq 3$ tend to be equal because there is no regularization on the other two β_i 's except the one with the maximum norm. Therefore, these DPMs tend to have equal margins and their outputs are more comparable. After the positive examples are assigned to different viewpoints, we train a full model for each viewpoint from which candidate occlusion patterns are generated. As the datasets do not provide occlusion annotations, we obtain fully visible examples for each viewpoint in a heuristic way: from its group, we simply select 70 percent of the positive examples which have the highest matching scores as fully visible examples. The full models of

FIGURE 4.7. Comparison of different approaches using HOG features on the car category in VOC2007.

different viewpoints are also trained jointly in a single SVM with the max-regularized objective function in Eq. (4.11). To make a fair comparison with DPM, the number of parts is fixed to 8 and the size of each part is set to 6×6 . For the three viewpoints, we generate a pool of $255 \times 3 = 765$ occlusion patterns. For each object category, we select at most 8 occlusion patterns, i.e. $K = 8$. $C = 0.003$ is used to train the detector ensemble using HOG features for the selected occlusion patterns.

4.4.2.2 Car detection results

We first compare our approach with DPM and And-Or [23] on the car category in the VOC2007 dataset. As ours, And-Or first obtains occlusion patterns and then train a detector ensemble using HOG features for object detection. Figure 4.7 provides a comparison between our approach and the two baselines. It can be seen that both And-Or and our approach outperforms DPM,

which demonstrates the advantage of occlusion handling. Furthermore, our approach (OCCL-D) outperforms And-Or by 1.5%. The And-Or approach mines occlusion patterns using synthetic 3D car data in which viewpoint and part-level occlusion information is available, while our approach only needs bounding-box annotations and is therefore more flexible. On the one hand, our approach gets a higher recall than And-Or. This is probably because the distribution of occlusion patterns in the synthetic data is not consistent with that in the real data. Some representative occlusion patterns may not be included in the And-Or model. Different from And-Or, our approach discovers occlusion patterns directly from the real data. On the other hand, And-Or has a higher precision when the recall is low. And-Or has part annotations and viewpoint information from the synthetic data for training, which probably results in a better model, while our approach takes part locations and viewpoints as latent variables. OCCL-D improves the performance over OCCL by 0.7%, showing the effectiveness of the discriminative power term for occlusion pattern selection. For the car category, occluder modeling does not improve the performance (OCCL-DO performs slightly worse than OCCL-D).

4.4.2.3 Performance analysis on partially occluded objects

We use a diagnostic tool [133] to analyze the performance of our approach on partially occluded objects. The tool provides occlusion level annotations for seven object categories in the PASCAL VOC2007 dataset: aeroplane, bicycle, bird, boat, cat, chair and diningtable. Each object is labeled by one of four occlusion levels: None, Low, Moderate and High. Detection performance is evaluated using average normalized precision (AP_N) instead of average precision (See [133] for details). Figure 4.8 shows the results of DPM and OCCL-D on the seven object categories. Table 4.2 summarizes the results of the two approaches on different occlusion levels. On the Moderate occlusion level, OCCL-D outperforms DPM on all categories except for the bird category. The improvement of OCCL-D over DPM is significant on cat and chair (9% and 5% respectively). On average, OCCL-D outperforms DPM by 3.4% over the seven object categories. On the Low occlusion level, OCCL-D outperforms DPM on 5 out of 7 object categories and its mean AP_N is 1.1% higher than that of DPM. On the Heavy occlusion level, OCCL-D

FIGURE 4.8. Results of DPM and OCCL-D on four occlusion levels. Black dashed lines indicate overall AP_N . Red bars indicate standard errors. The horizontal axis represents different occlusion levels: N = None, L = Low, M = Moderate and H = High.

occlusion level	method	aeroplane	bicycle	bird	boat	cat	chair	diningtable	mean
Moderate	DPM	13	35	0	10	10	8	42	16.9
	OCCL-D	16	37	0	13	19	13	44	20.3
Low	DPM	32	73	2	10	21	20	35	27.6
	OCCL-D	24	73	7	11	25	25	36	28.7
Heavy	DPM	0	44	0	0	21	3	39	15.3
	OCCL-D	0	51	0	0	1	3	36	13
None	DPM	42	68	7	21	28	22	28	30.9
	OCCL-D	50	68	12	25	33	26	16	32.9
All	DPM	39	65	5	17	25	17	37	29.3
	OCCL-D	44	65	10	20	30	21	38	32.6

TABLE 4.2. Average Normalized Precisions of DPM and OCCL-D on different occlusion levels. The All level includes all occlusion levels.

has a lower mean AP_N than DPM due to its extremely poor performance on the cat category. According to the results on the Moderate, Low and Heavy occlusion levels, the performance improvement of OCCL-D over DPM is more obvious on the Moderate occlusion level as most discovered occlusion patterns are close to this occlusion level (See Fig. 4.11 for some examples). From the analysis, we can see that our approach improves the detection performance on partially occluded objects for most categories, which demonstrates the effectiveness of occlusion handling of our approach. We also show the results of the two approaches on objects that are not occluded (None occlusion level) in Table 4.2. OCCL-D outperforms DPM on most object categories with an average improvement of 2.0%. On the All occlusion levels, OCCL-D outperforms DPM on all the seven object categories with an average improvement by 3.3%.

4.4.2.4 Results with varying numbers of occlusion patterns

We conduct experiments on the car and dog categories in VOC2007 with the number of occlusion patterns K ranging from 5 to 12. Figure 4.9(a) shows the results of OCCL-D on the car category. The performance of OCCL-D increases gradually when K changes from 5 to 8, but after that adding more occlusion patterns decreases its performance from 60.2% (P8) to 58.2% (P12). For the car category, about 8 occlusion patterns are enough for OCCL-D to work well. Using more occlusion patterns does not increase the recall but bring more false positives. Figure 4.9(b) shows the results of OCCL-D on the dog category. The performance improves when K increases from 5 to 11 and adding more occlusion patterns after $K = 11$ does not help. As dog examples are non-rigid and have more complex occlusions (e.g. self-occlusions), more occlusion patterns are needed to better represent them. Overall, OCCL-D works well with 8 occlusion patterns for 20 object categories in VOC2007 and VOC2010.

4.4.2.5 Results on 20 object categories

Table 4.3 shows the results of different approaches using HOG features on 20 object categories in the VOC2007 dataset. Both of our implementations (OCCL and OCCL-D) have a higher

FIGURE 4.9. Results of OCCL-D on the car and dog categories in the VOC2007 dataset. PK means K occlusion patterns are used. Numbers in the legend refer to average precisions.

	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table
DPM [10]	33.2	59.3	10.3	15.7	26.6	52.0	53.7	22.5	20.2	24.3	26.9
DPM-BB [10]	33.2	60.3	10.2	16.1	27.3	54.3	58.2	23.0	20.0	24.1	26.7
OCCL	37.1	59.6	11.9	18.3	33.2	50.9	59.5	23.2	23.2	29.2	25.8
OCCL-D	37.9	59.7	12.3	18.3	33.3	51.4	60.2	25.6	23.7	30.1	26.1
	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mean	
DPM [10]	12.6	56.5	48.5	43.3	13.4	20.9	35.9	45.2	42.1	33.2	
DPM-BB [10]	12.7	58.1	48.2	43.2	12.0	21.1	36.1	46.0	43.5	33.7	
OCCL	13.6	60.1	47.5	44.0	14.8	25.0	36.5	45.0	42.6	35.1	
OCCL-D	13.1	59.4	47.8	43.9	15.0	27.0	36.9	44.9	43.1	35.5	

TABLE 4.3. Average precision of different approaches using HOG features on 20 object categories in the VOC2007 dataset. DPM-BB uses bounding-box prediction to refine the results of DPM. Both OCCL and OCCL-D do not use any post-processing.

	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table
DPM [10]	42.9	47.2	10.3	11.1	26.3	48.4	40.2	22.9	17.0	22.9	10.2
DPM-BB [10]	45.6	49.0	11.0	11.6	27.2	50.5	43.1	23.6	17.2	23.2	10.7
OCCL	47.0	49.5	12.2	12.7	31.7	48.0	46.0	28.1	16.1	27.3	6.0
OCCL-D	46.9	49.5	13.1	12.7	32.0	48.3	45.8	28.3	17.0	26.6	4.4
	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mean	
DPM [10]	19.9	41.5	44.0	41.0	7.6	28.3	18.2	39.0	32.9	28.6	
DPM-BB [10]	20.5	42.5	44.5	41.3	8.7	29.0	18.7	40.0	34.5	29.6	
OCCL	21.0	42.8	44.4	46.1	12.3	29.7	20.1	33.0	32.3	30.3	
OCCL-D	21.7	43.1	45.5	45.9	11.2	30.7	20.3	34.2	37.5	30.7	

TABLE 4.4. Average precision of different approaches using HOG features on 20 object categories in the VOC2010 dataset. DPM-BB uses bounding-box prediction to refine the results of DPM. Both OCCL and OCCL-D do not use any post-processing.

mean AP than DPM and DPM-BB. Our approach performs best on 14 object categories and outperforms the other two approaches on several object categories with a large margin. For object categories like car and cow, our approach achieves a significant improvement since these object categories have small deformation and the heuristic method of initializing viewpoints and obtaining fully visible examples works well. In contrast, our approach performs poorly on object categories such as diningtable and train due to the difficulty of the initialization. OCCL-D achieves a higher mean AP than OCCL and outperforms OCCL on most object categories, which justifies the strategy that when two occlusion patterns are similar, the more discriminative one is preferred. Table 4.4 shows the results on the VOC2010 dataset. Our approach

FIGURE 4.10. Examples of discovered occlusion patterns and detections. The regions of a part in a detection and its corresponding occlusion pattern are marked with the same color. For clarity, only one typical detection is displayed in each image.

outperforms DPM and DPM-BB on 16 object categories and OCCL-D has a better overall performance than OCCL over 20 object categories. Note that both OCCL and OCCL-D do not use bounding-box regression [10] for refining detection results. Figure 4.10 gives some examples of discovered occlusion patterns and detections obtained by OCCL-D.

4.4.3 Experiments with Faster R-CNN

4.4.3.1 Implementation

We combine our approach with Faster R-CNN [15] and evaluate its performance on the VOC2007 and VOC2010 datasets. We fine-tune the ResNet-101 [81] on the *trainval* subsets of VOC2007 and VOC2012, i.e. the 07+12 setting in [14], using the Faster R-CNN code [15]. The fine-tuned ResNet-101 is used for testing on both VOC2007 and VOC2010. For each object category, we apply the ResNet-101 to collect about 400 candidate detections after non-maximum suppression in each image which are rescored by the learned non-linear SVM described in Section 4.3.3. The overlap threshold τ in Eq. (4.4) is set to 0.5 for matching candidate detections to the discovered occlusion patterns. For training the non-linear SVM, candidate detections which have an overlap greater than 0.7 with any ground-truth annotation are taken as positive examples and those which do not have an overlap greater than 0.3 with any ground-truth annotation are taken as negative examples. The parameter σ in Eq. (4.9) is selected for each object category using the *train* and *val* split for both VOC2007 and VOC2010. We compare our approach with Faster R-CNN [15] which uses the candidate detections from the ResNet-101.

4.4.3.2 Experimental results

Table 4.5 shows the results of Faster R-CNN and our approach on the VOC2007 dataset. Our implementation of Faster R-CNN with the ResNet-101 has a similar performance to the result reported in [81]: 76.7 vs 76.4 in mean AP (See FRCN+ResNet in Table 4.5). Our approach outperforms the baseline Faster R-CNN on 19 out of 20 object categories with an improvement of

	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table
Faster R-CNN	78.9	80.6	78.6	67.6	65.6	87.9	80.0	88.8	61.2	83.3	72.5
Ours	80.4	81.1	79.6	68.3	67.5	87.7	80.3	89.1	62.9	84.2	72.6
FRCN+ResNet [81]	79.8	80.7	76.2	68.3	55.9	85.1	85.3	89.8	56.7	87.8	69.4
	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mean	
Faster R-CNN	79.3	85.2	79.8	71.6	47.0	74.1	80.6	83.2	78.1	76.7	
Ours	79.4	86.8	80.2	73.2	48.8	74.2	82.3	86.5	79.0	77.7	
FRCN+ResNet [81]	88.3	88.9	80.9	78.4	41.7	78.6	79.8	85.3	72.0	76.4	

TABLE 4.5. Average precisions of Faster R-CNN and our approach on 20 object categories in the VOC2007 dataset. ResNet-101 is used for Faster R-CNN.

	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table
Faster R-CNN	79.6	78.4	75.0	60.4	60.7	74.2	70.9	89.5	53.6	71.0	59.0
Ours	81.1	79.6	77.2	60.7	59.3	74.8	69.7	89.8	54.0	74.2	59.4
	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mean	
Faster R-CNN	87.9	80.0	76.6	74.4	50.6	70.0	66.9	83.1	70.2	71.6	
Ours	88.3	82.1	80.1	75.3	50.1	74.6	70.3	83.3	70.1	72.7	

TABLE 4.6. Average precisions of Faster R-CNN and our approach on 20 object categories in the VOC2010 dataset. ResNet-101 is used for Faster R-CNN.

1.0% in mean AP. Table 4.6 shows the results of the two approaches on the VOC2010 dataset. Our approach has better performance than Faster R-CNN on 16 out of 20 object categories. The mean AP of our approach is 1.1% higher than that of Faster R-CNN. From the results on VOC2007 and VOC2010, we can see that the occlusion patterns obtained by our occlusion pattern discovery approach can be combined with Faster R-CNN to improve its performance. Besides performance improvement, the discovered occlusion patterns can also be used for occlusion reasoning as described in Section 4.3.3. Figure 4.11 shows some detection examples with occlusion reasoning. Note that besides Faster R-CNN, the discovered occlusion patterns can also be combined with state-of-the-art object detection approaches such as YOLO [38, 95] and SSD [98].

4.5 Conclusion

In this chapter, we propose a data-driven approach to discover occlusion patterns based on a classic part detector ensemble which is a DPM for partially occluded object detection. We

FIGURE 4.11. Detections with occlusion inferring. Red bounding boxes are detections obtained by Faster R-CNN. These detections are aligned with their matching occlusion patterns. Green and blue bounding boxes represent visible parts and estimated full objects respectively. Some images are padded with black regions to display estimated full objects. For clarity, only one typical detection is displayed in each image.

model occlusion pattern discovery as a facility location problem which can be solved effectively via a greedy algorithm. The approach applies to general object categories and is capable of discovering occlusion patterns that are both representative and discriminative. The discovered occlusion patterns after training can be applied directly for object detection as the classic deformable part models. They can also be combined with state-of-the-art detectors like Faster R-CNN for improving performance and inferring occlusions. The effectiveness of the approach

is validated on the PASCAL VOC2007 and VOC2010 datasets. We focus on discovering occlusion patterns of single objects for general categories in this paper and leave the extension to multi-object occlusion pattern discovery as explored in [24, 25, 44, 45] for future work.

Chapter 5

Learning to Integrate Part Detectors for Pedestrian Detection

5.1 Introduction

Pedestrian detection has many applications such as video surveillance and autonomous driving and much work has been done to improve its performance in recent years [11, 12, 32, 103, 108, 113]. Despite recent progress, partial occlusions are still a great challenge for pedestrian detection. Some state-of-the-art approaches achieve promising results on pedestrian detection benchmarks like Caltech [3] when pedestrians to be detected are not occluded or only slightly occluded. When heavy occlusions are present, their performances decrease drastically. For example, the log-average miss rate of Checkerboard detector [108] on the Caltech benchmark is 18.5% when only unoccluded or partially occluded pedestrians are considered. Its performance drops to 77.5% when pedestrians to be detected are heavily occluded. More efforts are still required to improve the performance of detecting heavily occluded pedestrians.

A simple yet effective approach to occlusion handling for pedestrian detection is to construct an ensemble of part detectors for various occlusion patterns [31, 32]. Occlusion patterns can be manually designed according to prior knowledge [31] or automatically selected from a large

FIGURE 5.1. Different integrations of part detectors. Left: The average pattern 0 is obtained by averaging visible parts of pedestrian examples in the Caltech dataset [3] and the top 4 occlusion patterns whose detectors have the highest weights are shown below. Blue regions in the occlusion patterns indicate visible parts of the human body. The pedestrian which is heavily occluded in the image below is ranked at 4th among top 5 regions. Right: Our approach clusters the pedestrian examples into 2 groups each of which selects its own compatible part detectors. The detectors associated with the average pattern 1 rank the pedestrian in the image below at 1st.

pool of candidates [32]. After occlusion patterns are obtained, a part detector is trained independently for each occlusion pattern. To make the approach work well, it is important to integrate the part detectors properly. Specifically, for a given image region, a detection score should be assigned to it based on the outputs of these part detectors. A simple method is to assign the maximum among these outputs to the image region as adopted in [31]. This method needs a sophisticated score calibration to make the outputs of the independently trained detectors comparable. In [32], a weight vector is learned by training a linear support vector machine (SVM) to combine the outputs of part detectors. The important part detectors would have large combination weights thus score calibration is not required. However, as this method only learns a single weight vector for all pedestrian examples, it may not be able to well separate these pedestrian examples from background especially when some of these examples are heavily occluded. A heavily occluded pedestrian usually receives low scores from the detectors of some occlusion patterns whose visible portion is incompatible with the visible part of the pedestrian.

For example, the part detector of the occlusion pattern in which only the lower body is visible would probably assign a low score to a pedestrian whose lower body is occluded. Therefore, with the single weight vector, heavily occluded pedestrians tend to have a relatively low detection score compared to fully visible pedestrians and are not easy to be distinguished from the background as illustrated in the left of Fig. 5.1. In addition, some part detectors, especially those unreliable ones, tend to produce noisy outputs and including them may not help improve and even decrease the detection performance.

Following the framework in [32], we propose a new approach to integrate part detectors for heavily occluded pedestrian detection. Instead of only learning a single weight vector to fuse all part detectors, we cluster pedestrian examples into several groups according to their occlusion patterns and learn a weight vector for each group such that a pedestrian example in the group is scored by the linear combination of the outputs of the part detectors specified by the weight vector. To remove irrelevant and unreliable part detectors, we impose sparsity on the weight vector of each group. The weight vectors are learned simultaneously by training an L1-norm linear SVM [139]. Our integration approach selects highly compatible part detectors for each group and can better distinguish heavily occluded pedestrians from the background as illustrated in the right of Fig. 5.1. The effectiveness of our approach is demonstrated on the Caltech dataset [3].

The remainder of this chapter is organized as follows. Section 5.2 describes how to train part detectors for occlusion patterns. The proposed part detector integration approach is presented in Section 5.3. Experimental results are given in Section 5.4. Section 5.5 concludes the whole chapter.

5.2 Training Part Detectors

Occlusions may occur at different parts of a pedestrian and have various patterns. For example, the lower body of a pedestrian may be occluded by a car, and the left or right half body may be occluded by a pole or another pedestrian. As in [32], we construct a pool of occlusion patterns

FIGURE 5.2. Examples of occlusion patterns. (a) The right portion of the human body is occluded. Two instances marked by red rectangles are shown on the right: one is occluded by another pedestrian and the other is truncated by the image boundary. (b) The lower body is occluded. Two pedestrians on the right are occluded by cars.

in which different parts of a human body are occluded. The human body is represented by an $R \times C$ grid, where R and C are the numbers of cells in one row and one column respectively. We sample all possible rectangular subregions of size $r \times c$ in the grid with $R_{min} \leq r \leq C$ and $C_{min} \leq c \leq C$. Each subregion together with the grid form one occlusion pattern in which the subregion represents the visible part of the human body. To avoid sampling subregions that are too small, we restrict the minimum height and width of a subregion. For each size $r \times c$, we sample subregions from top-left to bottom-right in the grid with a step length of one cell in both horizontal and vertical directions. Mathematically, an occlusion pattern can be represented by a 4-tuple $p = (x, y, r, c)$, where (x, y) are the top-left coordinates of the occlusion pattern in the grid and (r, c) specifies its width and height. We set $R_{min} = 2$, $C_{min} = 2$, $R = 6$ and $C = 3$ in our experiments, resulting in a pool of 45 occlusion patterns. The occlusion pattern pool can be expressed as $P = \{p_i | 1 \leq i \leq M\}$, where $p_i = (x_i, y_i, r_i, c_i)$ and $M = 45$. Figure 5.2 shows two examples of occlusion patterns.

For each occlusion pattern, we train a detector using locally decorrelated channel features (LDCF) [107] and RealBoost [140] to detect the visible part in the occlusion pattern. Figure 5.3 illustrates the training process for the upper-body pattern. LDCF belongs to a family of channel features [11, 103, 104, 108] which are widely used for pedestrian detection. We consider the human body as a rigid object and model it with a template of $H \times W$ pixels,

FIGURE 5.3. Process of training a detector for the upper-body pattern.

where H and W are the height and width, respectively. As the ground-truth bounding boxes of positive examples usually have different sizes and aspect ratios, we perform bounding box standardization as in [3]: for each ground-truth bounding box, we adjust its width such that the aspect ratio after adjustment is equal to $\frac{H}{W}$ while keeping its height and center coordinates unchanged. An example of bounding box standardization is given at the top-left of Fig. 5.3. The original bounding box is marked in red and the green rectangle is the new bounding box after standardization. It is pointed out in [12] that including a certain amount of background around a pedestrian as context can improve detection performance. To exploit the surrounding context, we add some padding to the template boundary. Denote the height and width of the padded template by H' and W' respectively. Accordingly, we expand the standardized bounding box of a positive example by including some background around it. The yellow rectangle at the top-left of Fig. 5.3 shows the expanded bounding box of the pedestrian example. Finally, we crop the region corresponding to the visible part of the occlusion pattern from the expanded bounding box as a positive training patch. Negative patches are sampled from background regions. For feature extraction, each training patch is scaled to the size of the visible part in the template. For example, the size of a training patch for the upper-body pattern is $\frac{H'}{2} \times W'$. We use the same channels as in [11] to compute features for training examples: normalized

gradient magnitude (1 channel), LUV color channels (3 channels) and histograms of oriented gradients (6 channels), with a total of 10 feature channels for each training patch. Each feature channel is a per-pixel lookup table which has the same size as the training patch. We further downsample the feature channels by a factor of 2. LDCF learns a set of k $m \times m$ filters to locally decorrelate channel features where k is a predefined number (See [107] for details). Applying these k filters to the 10 features channels results in a total of $10k$ feature channels. We downsample the $10k$ feature channels by a factor of 2 to represent the training patch. If the size of a training patch is $s \times t$ ($s = \frac{H'}{2}$ and $t = W'$ for the upper-body pattern), the final feature representation has $\frac{s}{4} \times \frac{t}{4} \times 10k$ dimensions. With a set of training patches, we train and combine a set of weak classifiers which are decision trees in our implementation to obtain a boosted part detector using RealBoost.

Let L be the number of weak classifiers of each part detector and d_i be the part detector of the i -th occlusion pattern p_i . Given an image patch x whose size is $H' \times W'$, we denote by $\psi_i(x)$ the channel features extracted from x corresponding to p_i . The detection score of x given by d_i is computed by

$$d_i(x) = \sum_{j=1}^L w_i^j f_i^j(\psi_i(x)), \quad (5.1)$$

where f_i^j is the j -th weak classifier of d_i and w_i^j is the corresponding weight. The set of part detectors trained for the occlusion patterns in P can be expressed as $D = \{d_i | 1 \leq i \leq M\}$.

5.3 Integrating Part Detectors

After obtaining the part detectors as described in Section 5.2, we need to integrate these part detectors properly for pedestrian detection. Specifically, given an image patch x whose size is $H' \times W'$ pixels, we want to assign a score $g(x)$ indicating how likely the image patch x contains a pedestrian based on the detection scores $d_i(x)$ ($1 \leq i \leq M$) from the part detectors in D .

5.3.1 Linear Integration

In [32], a linear SVM is trained to combine detection scores from different part detectors which are learned by deep neural networks. This method also applies to our part detectors. Mathematically, the goal is to learn a weight vector $\mathbf{a} = [a_1, \dots, a_M]$ to linearly combine the detection scores from the M part detectors

$$g(x) = \sum_{i=1}^M a_i d_i(x) + b, \quad (5.2)$$

where b is a constant. Given a set of training examples (x_i, y_i) for $1 \leq i \leq N$ with x_i an image patch and $y_i \in \{1, -1\}$ the label of x_i , the weight vector $\mathbf{a}$ is obtained by solving the following optimization problem:

$$\min_{\mathbf{a}, b} \|\mathbf{a}\|_2^2 + C \sum_{i=1}^N \max(0, 1 - y_i g(x_i)), \quad (5.3)$$

where C is a parameter. By defining a score vector $\mathbf{s} = [d_1(x), \dots, d_M(x)]$, we can write the score of x as $g(x) = \langle \mathbf{a} \cdot \mathbf{s} \rangle + b$, where $\langle \mathbf{a} \cdot \mathbf{s} \rangle$ is the inner product of $\mathbf{a}$ and $\mathbf{s}$.

Nevertheless, this method has two limitations. First, it only learns a single weight vector for all pedestrian examples with various occlusion patterns. The decision boundary specified by $\mathbf{a}$ and b may not well separate these pedestrian examples from the background, especially when heavy occlusions are present. Actually, for a heavily occluded pedestrian, some part detectors are irrelevant and tend to give a low detection score. For example, an occlusion pattern in which the upper body is occluded does not have any overlap with a pedestrian whose lower body is occluded thus the pedestrian would probably receive a low detection score from the part detector of that occlusion pattern. Therefore, heavily occluded examples usually have a relatively low score compared with fully visible pedestrians and are not easy to be distinguished from the background (See the left part of Fig. 5.1). Second, some part detectors are not reliable and may produce noisy outputs. Including these part detectors may not help improve and even decrease the detection performance.

FIGURE 5.4. Visible part clustering. (a) Red and green bounding boxes show the full body and visible part of a pedestrian example respectively. (b) Four cluster centers obtained by K-means on pedestrian examples in the Caltech dataset. Cyan and white regions indicate visible and occluded parts respectively.

5.3.2 Our Approach

To address the two limitations of linear integration, we cluster pedestrian examples into several groups according to how they are occluded and learn a weight vector for each group such that only a subset of part detectors which are both relevant and reliable would contribute to the group.

Each pedestrian example is annotated with two bounding boxes which denote the visible part $B_{vis} = (x_{vis}^1, y_{vis}^1, x_{vis}^2, y_{vis}^2)$ and the full body $B_{full} = (x_{full}^1, y_{full}^1, x_{full}^2, y_{full}^2)$ respectively (See Fig. 5.4(a)). Let $w_{full} = x_{full}^2 - x_{full}^1 + 1$ and $h_{full} = y_{full}^2 - y_{full}^1 + 1$ be the width and height of B_{full} respectively. We normalize the visible part B_{vis} relative to the full body B_{full} to obtain a new bounding box $B_{norm} = (x_{norm}^1, y_{norm}^1, x_{norm}^2, y_{norm}^2)$, where

$$x_{norm}^1 = \frac{x_{vis}^1 - x_{full}^1}{w_{full}}, y_{norm}^1 = \frac{y_{vis}^1 - y_{full}^1}{h_{full}}, x_{norm}^2 = \frac{x_{vis}^2 - x_{full}^1}{w_{full}}, y_{norm}^2 = \frac{y_{vis}^2 - y_{full}^1}{h_{full}}. \quad (5.4)$$

Then, we use K-means to cluster pedestrian examples into K groups according to their normalized visible parts. Figure 5.4(b) shows the clustering result on pedestrian examples in the Caltech dataset [3] with $K = 4$. It can be seen that pedestrian examples in the Caltech dataset

are usually occluded from the left, right or bottom.

After clustering the pedestrian examples into K groups, we learn a weight vector $\mathbf{c}_j = [c_j^1, \dots, c_j^M]$ and a bias b_j for each group with $1 \leq j \leq K$ such that a pedestrian example x in the j -th group is scored by

$$g_j(x) = \sum_{i=1}^M c_j^i d_i(x) + b_j. \quad (5.5)$$

However, if we learn the weight vectors separately for each group, the final scores may not be comparable and a further calibration would be required. To solve this problem, we propose to learn $\mathbf{c}_j$ and b_j for $1 \leq j \leq K$ simultaneously by training a single L1-norm linear SVM [139]. The pedestrian examples comprise the positive training set and negative training examples are collected from background regions. Each training example x_i for $1 \leq i \leq N$ is labeled by $l_i = (y_i, g_i)$ where $y_i \in \{-1, 1\}$ indicates whether x_i is a pedestrian and $g_i \in \{1, \dots, K\}$ is its group index. Let $\mathbf{s}_i = [d_1(x_i), \dots, d_M(x_i)]$ be the detection scores of x_i from the M part detectors. We represent x_i by a long feature vector $\mathbf{u}_i = [\mathbf{u}_i^1, \dots, \mathbf{u}_i^K]$ where $\mathbf{u}_i^j$ ($1 \leq j \leq K$) is a $(M+1)$ -dimensional feature vector corresponding to the j -th group and is defined by

$$\mathbf{u}_i^j = \begin{cases} [\mathbf{s}_i, B] & \text{if } g_i = j; \\ [\mathbf{0}, B] & \text{otherwise,} \end{cases} \quad (5.6)$$

where B is a constant feature. We learn a long weight vector $\mathbf{w} = [\mathbf{w}^1, \dots, \mathbf{w}^K]$, where $\mathbf{w}^j = [\mathbf{w}^j(1), \dots, \mathbf{w}^j(M+1)]$ ($1 \leq j \leq K$) is a $(M+1)$ -dimensional weight vector corresponding to the j -th group, by solving the following optimization problem:

$$\min_{\mathbf{w}} \|\mathbf{w}\|_1 + C \sum_{i=1}^N \max(0, 1 - y_i \langle \mathbf{w} \cdot \mathbf{u}_i \rangle). \quad (5.7)$$

The lasso penalty $\|\mathbf{w}\|_1$ is first proposed in [141] for regression problems and its L1 nature would cause some weights in $\mathbf{w}$ to be exactly zero when C is sufficiently small (See [139] for more discussions). For a specific group, irrelevant and unreliable part detectors are usually less important and tend to receive a weight of zero. After $\mathbf{w}$ is learned, we can obtain $\mathbf{c}_j$ and b_j by setting $\mathbf{c}_j = [\mathbf{w}^j(1), \dots, \mathbf{w}^j(M)]$ and $b_j = B\mathbf{w}^j(M+1)$. We choose a large value for B to

reduce the impact of its corresponding weights $\mathbf{w}^j(M+1)$ for $1 \leq j \leq K$ in the lasso penalty $\|\mathbf{w}\|_1$ ($B = 5000$ is used in our experiments). At testing stage, the score of an image patch x is determined by the largest score from the K groups:

$$g(x) = \max_{1 \leq j \leq K} g_j(x). \quad (5.8)$$

5.4 Experiments

We evaluate the proposed approach on the Caltech dataset [3]. Following the standard evaluation protocol, we use video sets S0-S5 for training and S6-S10 for testing. Detection performance is summarized by log average miss rate over 9 false positive per-image (FPPI) points ranging from 10^{-2} to 10^0 . Three subsets from S6-S10 are used for testing: *Reasonable*, *Partial* and *Heavy*. In the *Reasonable* subset, only pedestrians with at least 50 pixels tall and under no or partial occlusion are used for evaluation. This subset is widely used for evaluating pedestrian detection approaches. In the *Partial* and *Heavy* subsets, pedestrians are at least 50 pixels tall and are partially occluded (1-35 percent occluded) and heavily occluded (35-80 percent occluded) respectively.

5.4.1 Implementation Details

For training the part detectors for occlusion patterns, we set the template size of the human body to 100×41 and the template size becomes 112×48 after an amount of padding is added. Four 5×5 filters are used for LDCF extraction, producing a total of 40 feature channels for an image patch. We sample training data from video sets S0-S5 at an interval of 2 frames. The maximum depth of a decision tree is set to 5 and a boosted part detector of $L = 4096$ decision trees is trained for each occlusion pattern. We adopt five rounds of bootstrapping to learn 64, 512, 1024, 2048 and 4096 decision trees respectively.

FIGURE 5.5. (a) Results of part detectors. (b) Results of our LDCF implementation and some state-of-the-art approaches using channel features.

For integrating part detectors, we sample pedestrians that are at least 50 pixels tall and are occluded no more than 60 percent as positive examples. Negative examples are collected by several rounds of hard mining. We use LIBLINEAR [142] and G-SVM [143] for solving linear SVMs and L1-norm linear SVMs respectively. Besides linear combination, we also implement a max integration approach in which the final score of an example x is defined as $g(x) = \max_{1 \leq i \leq M} d_i(x)$. We convert the output of each part detector d_i into a probability by logistic regression:

$$d'_i(x) = \frac{1}{1 + \exp(-q_i d_i(x) - r_i)}, \quad (5.9)$$

where q_i and r_i are the parameters to be learned. Then, the final score becomes $g'(x) = \max_{1 \leq i \leq M} d'_i(x)$.

5.4.2 Results of Part Detectors

Figure 5.5(a) shows the results of part detectors of some common occlusion patterns given in Fig. 5.6 on the *Reasonable* subset. We can see that the log-average miss rates of these part detectors vary largely from 18.5% to 61.2%. The part detector of the full body pattern P1 performs best among the seven part detectors. The part detectors of P2 and P3 performs slightly worse than the part detector of P1. From P4 to P7, the performance increases gradually when

FIGURE 5.6. Common occlusion patterns.

the amount of occlusion decreases. Generally, the part detector of a heavily occluded pattern performs worse than the part detector of a pattern that is only slightly or not occluded on the *Reasonable* subset. Figure 5.5(b) shows the comparison between our implementation of LDCF (LdcfP1) and some state-of-the-art approaches which are also based on channel features: ACF [11], LDCF [107], SpatialPooling+ [106] and Checkerboards [108]. LdcfP1 achieves a much better performance than the original LDCF and performs as well as Checkerboards.

5.4.3 Results of Different Integration Methods

We compare the proposed integration approach with three baselines: Max, MaxCal and Linear. Linear is the approach proposed in [32] which learns a weight vector to linearly combine outputs of part detectors as described in Section 5.3. Max and MaxCal are two approaches using the max integration as described in 5.4.2 without and with score calibration respectively. For each baseline, we experiment with different numbers of part detectors. We set the number of part detectors m to 1, 5, 10, 15, 20, 25, 30, 35, 40 and 45 respectively. When $m = 1$, the part detector of the full body pattern is used. For $m > 1$, we choose top m part detectors with the highest combination weights learned by Linear. Figure 5.7 shows the top 20 occlusion patterns of Linear. For our approach, we adjust the parameter C in Eq. (5.7) to control the number of part detectors to be selected.

Table 5.1 shows the results of the three baselines and our approach applied to different numbers of part detectors on the *Reasonable*, *Partial* and *Heavy* subsets. It can be seen that MaxCal

FIGURE 5.7. Top 20 occlusion patterns of Linear with associated combination weights.

outperforms Max for different values of m on all the three subsets. As our part detectors are trained independently, proper score calibration is necessary for the max integration approach to work well. On the *Reasonable* subset, integrating a number of part detectors ($m > 1$) using MaxCal does not help improve the performance compared to the part detector of the full body pattern ($m = 1$). This is probably because the full-body detector already works reasonably well for detecting pedestrians which are slightly or not occluded, while for MaxCal, part detectors could introduce more false positives. When the amount of occlusion increases, the other part detectors can complement the full-body detector for detecting partially and heavily occluded pedestrians. As shown in Table 5.1, the best performance of MaxCal is achieved when $m > 1$ on both *Partial* and *Heavy* subsets. Linear outperforms MaxCal for most values of m on the three subsets. In Linear, the outputs of part detectors are adjusted with combination weights and the complementarity among different part detectors are exploited to better distinguish pedestrians from background. In most cases, integrating a number of part detectors using Linear outperforms a single part detector as shown in Table 5.1. The best performances of Linear on the three subsets are achieved at $m = 20$, $m = 35$ and $m = 40$ respectively. Overall, our approach achieves better performance than Linear for most values of m on the three subsets. According to the results of Linear and Ours in Table 5.1, more part detectors are needed with the increase of occlusion. Note that the best performance of Linear and Ours on each testing subset is not achieved at $m = 45$. This shows that including all the part detectors does not

FIGURE 5.8. Average occlusion patterns and top occlusion patterns of two groups of Ours-g2.

necessarily have the best performance as some part detectors can produce noisy outputs.

Figure 5.9 shows the results of the proposed integration approach and three baselines: Max-m45, MaxCal-m45 and Linear-m45. All the three baselines use $m = 45$ part detectors. Ours-g K is our approach with K groups. It can be seen that our approach performs best on all the three subsets. The difference between Linear-m45 and Ours-g1 is that the combination weights of Linear-m45 is learned by a linear SVM, while Ours-g1 adopts an L1-norm SVM to obtain its combination weight vector. With the sparsity constraint, Ours-g1 only selects 37 part detectors and discards the remaining 8 noisy ones. Ours-g1 outperforms Linear-m45 consistently on

	$m=1$	$m=5$	$m=10$	$m=15$	$m=20$	$m=25$	$m=30$	$m=35$	$m=40$	$m=45$
Max	18.5	25.5	26.0	24.8	23.5	26.6	28.6	29.1	29.1	29.2
MaxCal	18.5	19.0	20.2	20.2	18.8	18.8	18.6	19.1	18.9	18.5
Linear	18.5	17.4	21.2	16.4	15.6	17.3	16.8	16.0	16.3	16.2
Ours	18.5	15.9	15.8	15.4	15.3	15.3	15.3	15.3	15.5	16.0

(a) *Reasonable*

	$m=1$	$m=5$	$m=10$	$m=15$	$m=20$	$m=25$	$m=30$	$m=35$	$m=40$	$m=45$
Max	39.3	48.5	45.7	43.3	42.2	46.1	48.5	48.7	45.9	48.5
MaxCal	39.3	42.7	43.1	39.6	39.0	38.1	37.5	38.4	36.9	37.8
Linear	39.3	40.4	41.4	37.5	37.2	36.1	38.0	35.0	36.0	38.4
Ours	39.3	38.0	37.2	37.8	36.6	36.1	36.6	36.3	36.4	37.5

(b) *Partial*

	$m=1$	$m=5$	$m=10$	$m=15$	$m=20$	$m=25$	$m=30$	$m=35$	$m=40$	$m=45$
Max	75.7	72.3	78.2	78.3	77.9	76.8	73.8	73.4	73.1	73.4
MaxCal	75.7	72.0	74.7	74.6	74.5	74.0	72.6	73.1	72.9	72.4
Linear	75.7	77.0	71.9	74.6	74.2	73.0	79.2	75.3	71.5	75.0
Ours	75.7	75.0	74.6	74.4	74.4	73.8	72.8	72.7	72.5	74.0

(c) *Heavy*

TABLE 5.1. Results of different integration methods. The bold number in each row indicates the best performance of the corresponding method.

the *Reasonable*, *Partial* and *Heavy* subsets. Ours-g2 achieves a slightly worse performance than Ours-g1 on the *Reasonable* subset, but when the occlusion becomes more severe, Ours-g2 outperforms Ours-g1. Compared to Ours-g1, Ours-g2 achieves performance gains of 0.4% and 2.2% respectively on the *Partial* and *Heavy* subsets. The performance improvements of Ours-g2 over Linear-m45 on the three subsets are 0.9%, 2.7% and 4.9% respectively. The advantage of our approach over Linear-m45 becomes more obvious when heavy occlusions are present.

Figure 5.10 shows the results of our approach with different numbers of groups ranging from 1 to 5. We can see that from $K = 2$, increasing the number of groups does not help much for the detection performance. Actually, most pedestrians in the Caltech dataset are occluded from the bottom according to the occlusion statistics in [3]. Figure 5.8 shows the average occlusion patterns and top occlusion patterns with the highest combination weights of two groups of Ours-g2. The first group corresponds to pedestrians which are slightly or not occluded and most pedestrians in the second group are occluded from the bottom. Ours-g2 gives high weights to the full-body pattern and partially occluded patterns in the first group (See the first 5 occlusion

FIGURE 5.9. Results of our integration approach and three baselines.

patterns), while occlusion patterns in which the human body is occluded from the bottom are more important in the second group, for example, the first three occlusion patterns.

Figure 5.11 shows the results of four integration approaches and some state-of-the-art approaches. Ours-g2 achieves the best performance among approaches using channel features on the three subsets. Compared with our own implementation of LDCF LdcfP1, Ours-g2 achieves performance gains of 3.0%, 3.6% and 5.6% respectively on the *Reasonable*, *Partial* and *Heavy* subsets. Linear-m45 also outperforms LdcfP1 consistently on the three subsets. Generally, properly integrating a number of part detectors can achieve a better performance than the single full-body detector, especially when heavy occlusions are present. Currently, approaches using deep neural networks [32, 112] or features learned by these networks [111, 113] achieve better performance. Our part detector integration approach still applies when the part detectors

FIGURE 5.10. Results of our integration approach with different numbers of groups.

are replaced with part detectors learned by these approaches. We also compare our approach with some new approaches published after our work, MS-RCNN [92], RPN+BF [114], F-DNN [144], F-DNN+SS [144], DeepParts [32], CompACT-Deep [113], SDS-RCNN [145] and PCN [146] in Fig. 5.12.

5.5 Conclusion

In this chapter, we propose a new approach to improve the integration module of the framework of learning a part detector ensemble for detecting heavily occluded pedestrians. Instead of linearly combining all part detectors into a generic detector for all occlusions, we categorize occlusions based on how pedestrian examples are occluded into K groups each of which selects

FIGURE 5.11. Results of four integration approaches and some state-of-the-art approaches. Solid lines represent channel-feature based approaches and dashed lines represent deep-learning based approaches.

its own part detectors and fuses them linearly to obtain a classifier. The K classifiers are learned simultaneously by an L1-norm linear SVM which can effectively remove irrelevant and unreliable part detectors for each group. The experiments on the Caltech dataset demonstrate the effectiveness of our approach.

FIGURE 5.12. Results of our approach and new state-of-the-art approaches.

Chapter 6

Multi-label Learning of Part Detectors for Pedestrian Detection

6.1 Introduction

As discussed in Chapter 5, learning an ensemble of part detectors is a common solution to occlusion handling for pedestrian detection [31–36, 109]. When a full-body detector fails to detect a partially occluded pedestrian, the detectors of the parts which are still visible may give high detection scores (See Fig. 6.1). For this solution, the reliability of part detectors is of great importance, since part detectors are its building blocks. Usually, part detectors are learned independently [31–36]. This way of learning part detectors has two drawbacks: (1) Correlations among parts are ignored during learning, which would affect the reliability of the learned part detectors; (2) The computational cost of applying a set of part detectors increases linearly with the number of parts. In [109], part detector learning and integration are done in a single convolutional neural network. However, this approach only uses class labels and part detectors are learned implicitly in a weakly supervised fashion. We believe part-level supervision can be exploited to further improve its performance.

To address the above issues of learning part detectors, we propose a multi-label learning approach to jointly learn part detectors with part-level supervision. The goal of joint learning is

FIGURE 6.1. Occlusion handling. (a) Top five detections of a full-body detector. The heavily occluded pedestrian (Blue bounding box) is only ranked at 5th. (b) Detection scores of 20 part detectors on the five detections in (a). The first detector is the full-body detector. Figure 6.2 shows the 20 parts. (c) The five detections in (a) are re-ranked by properly integrating the 20 part detectors. The heavily occluded pedestrian (Blue bounding box) is ranked at 2nd.

to (1) improve the part detectors by exploiting correlations among parts, e.g. some parts of the body tend to appear/disappear together, and (2) reduce the computational cost of applying the learned part detectors for pedestrian detection. Learning and combining a set of decision trees via boosting to form a boosted forest is widely adopted for pedestrian detection and has shown to work well in practice [11, 108, 114, 118]. We also choose decision trees to form our part detectors. However, instead of learning a set of decision trees for each part detector, we only construct one set of decision trees which are shared by all the part detectors. To exploit part

correlations, these decision trees are learned and combined to capture the overall distribution of all the parts. We adapt AdaBoost.MH [147], which is a multi-class, multi-label version of AdaBoost, to learn these decision trees. For occlusion handling, we explore several methods to integrate the part detectors jointly learned by the proposed approach. We also exploit context to further improve the performance. The effectiveness of our approach is demonstrated on the Caltech [3] and CUHK [35] datasets. We apply the proposed multi-label learning approach to channel features [107] and features learned by a convolutional neural network (CNN features) respectively. When used for pedestrian detection individually, the part detectors learned jointly by the proposed approach show better performance than their counterparts learned separately. Using channel features our approach improves state-of-the-arts for detecting pedestrians in different occlusion situations while using CNN features our approach shows comparable performance for detecting pedestrians that are non-occluded or slightly occluded and achieves the best performance for detecting occluded pedestrians, especially heavily occluded ones.

In summary, our contributions are two-fold: (1) we propose a multi-label learning approach to jointly learn part detectors which share decision trees to exploit correlations among parts and also reduce the computational cost of applying these part detectors; (2) we explore several integration methods to integrate the part detectors learned by the proposed approach for occlusion handling and exploit context to further improve detection performance. The remainder of this chapter is organized as follows. Section 6.2 presents the proposed multi-label part detector learning approach. How to exploit the learned part detectors for occlusion handling is described in Section 6.3. Experimental results are given in Section 6.4. Section 6.5 concludes the whole chapter.

6.2 Multi-label Learning of Part Detectors

6.2.1 Part representation

We model the whole body of a pedestrian as a rectangular template without distinguishing different viewpoints. The template is divided into an $H \times W$ grid. Any rectangular sub-region in the template is considered as a part. Mathematically, a part can be expressed as a 4-tuple $p = (l, t, r, b)$, where (l, t) and (r, b) are the coordinates of the top-left and bottom-right corners of the part respectively with $0 \leq l < r \leq W$ and $0 \leq t < b \leq H$. In our implementation, we set $H = 6$ and $W = 3$. According to prior knowledge that pedestrians are usually occluded from the left, right or bottom, we manually design a pool of parts as shown in Figure 6.2. The part pool can be expressed as $\mathcal{P} = \{p_k | 1 \leq k \leq K\}$, where $p_k = (l_k, t_k, r_k, b_k)$ and $K = 20$. For pedestrian detection, images are usually represented by a set of feature maps, e.g. locally decorrelated channel features (LDCF) [107] and convolutional channel features (CCF) [111]. To represent a part on a pedestrian, a natural way is to crop regions that correspond to the part from the feature maps. One drawback of this representation is that small parts on a pedestrian are difficult to be reliably detected as the information from the small parts is relatively limited compared with that from large parts, especially when the pedestrian is small. Instead, we represent a part using features from the whole body. Features from the surrounding region of the part can be taken as its context. In this way, all the parts are represented by the same features.

6.2.2 Multi-label formulation

Let $\mathcal{X}$ be an instance space which consists of image regions. For each part $p_k \in \mathcal{P}$, we want to learn a detector $d_k : \mathcal{X} \rightarrow \mathbb{R}$ such that for an image region $x \in \mathcal{X}$, $d_k(x) > 0$ if the image region contains p_k and $d_k(x) \leq 0$ otherwise. A direct solution is to learn the K part detectors separately. However, this solution ignores correlations among the parts. For example, according to the part definition in Section 6.2.1, if a part appears in an image region, any smaller

FIGURE 6.2. Part pool. Blue regions denote parts. The first part is the whole body which is modeled as a template of 6×3 cells. Parts 2 to 17 are designed to handle situations where occlusions occur from the left, right or bottom and the last three parts are used for detecting the lower body.

part within this part should also be visible. To exploit potential correlations among the parts, we propose a multi-label learning approach to learn the part detectors jointly.

Specifically, we learn a function $F : \mathcal{X} \rightarrow 2^{\mathcal{P}}$ to predict an arbitrary set of parts $P \subseteq \mathcal{P}$ which appear in an given image region x . Let $\mathcal{D} = \{(x_i, l_i, B_i^v, B_i^f) | 1 \leq i \leq N\}$ be a set of training examples, where $x_i \in \mathcal{X}$ is an image region, $l_i \in \{-1, 1\}$ indicates whether the image region contains a pedestrian and if so, B_i^v and B_i^f are two bounding boxes specifying the visible portion and full body of the pedestrian respectively. To learn F , we need to construct for each instance x_i a label vector $\mathbf{Y}_i = (y_{ik}) \in \{-1, 1\}^K$ for $1 \leq k \leq K$, where y_{ik} indicates whether the image region x_i contains the part p_k . When a part is only partially visible on a pedestrian, it is not trivial to assign 1 or -1 to the part. Wrong assignment of part labels may cause the learning of part detectors to fail. So, we introduce a cost vector $\mathbf{C}_i = (c_{ik}) \in \mathbb{R}^K$ for $1 \leq k \leq K$ to soften the label assignment, where c_{ik} ($0 \leq c_{ik} \leq 1$) defines the cost incurred if a wrong prediction is made on x_i for p_k . For $l_i = -1$, we set $y_{ik} = -1$ and $c_{ik} = 1$. For $l_i = 1$, we set $y_{ik} = 1$ and determine the cost c_{ik} based on B_i^v and B_i^f . We first standardize the full-body bounding box B_i^f as in [3]: the bounding-box is adjusted such that after the adjustment the ratio of its width to its height is 0.41 with its height and center coordinates unchanged. Denote by R_i^f the standardized full body. Then, any image contents inside the visible portion B_i^v but outside the standardized full body R_i^f are discarded to obtain a new visible portion R_i^v .

FIGURE 6.3. Example labeling. The blue and red bounding boxes are the visible portion and full body of the pedestrian example respectively. The green bounding box is the standardized full body and the yellow bounding box is the new visible portion inside the standardized full body. The purple bounding box shows the image region of the upper-body part on the pedestrian example. The cost vector is calculated based on the purple and yellow bounding boxes.

Let R_{ik}^p be the image region of the part p_k on the instance x_i . We calculate the intersection over union (IOU) between R_{ik}^p and R_i^v denoted by I_{ik} , and the proportion of R_{ik}^p covered by R_i^v denoted by O_{ik} . Finally, the cost c_{ik} is determined as follows:

$$c_{ik} = \begin{cases} O_{ik} & O_{ik} \geq 0.7; \\ I_{ik} & O_{ik} < 0.7 \text{ and } I_{ik} \geq 0.5; \\ 0 & \text{otherwise.} \end{cases} \quad (6.1)$$

In the first case, the majority of the part p_k is visible on the instance x_i , so a large cost is set to prevent the part from being wrongly predicted. In the second case, the IOU between the part and visible portion is over 0.5. We thus consider the part to be visible on the instance and the cost of wrongly classifying it depends on the IOU. In the third case, the part is largely occluded. We set $c_{ik} = 0$ to discard this training example for the k -th part. Figure 6.3 illustrates how a pedestrian example is labeled. $\mathcal{D}_F = \{(x_i, \mathbf{Y}_i, \mathbf{C}_i) | 1 \leq i \leq N\}$ forms the training set for learning F . We identify F with a two-argument function $H : \mathcal{X} \times \mathcal{P} \rightarrow \mathbb{R}$ such that $p_k \in F(x)$

if $H(x, p_k) > 0$ and $p_k \notin F(x)$ otherwise. For any predicate π , let $[\pi]$ be 1 if π holds and 0 otherwise. We learn H by minimizing the following loss function:

$$L(H, \mathcal{D}_F) = \frac{1}{N} \sum_{i=1}^N \sum_{k=1}^K c_{ik} [\text{sign}(H(x_i, p_k)) \neq y_{ik}], \quad (6.2)$$

where $\text{sign}(H(x_i, p_k)) = 1$ if $H(x_i, p_k) > 0$ and $\text{sign}(H(x_i, p_k)) = -1$ otherwise.

6.2.3 Learning via boosting

Since the combination of boosting and decision trees has shown promising performance for pedestrian detection [11, 108, 114, 118], we choose decision trees to form our part detectors. We consider two approaches to minimizing the loss function $L(H, \mathcal{D}_F)$ in Eq. (6.2) for learning the part detectors.

The first approach learns the part detectors separately. Define the training loss related to the k -th part detector by

$$L_k(H, \mathcal{D}_F) = \frac{1}{N} \sum_{i=1}^N c_{ik} [\text{sign}(H(x_i, p_k)) \neq y_{ik}]. \quad (6.3)$$

$L(H, \mathcal{D}_F)$ can be decomposed as

$$L(H, \mathcal{D}_F) = \sum_{k=1}^K L_k(H, \mathcal{D}_F). \quad (6.4)$$

$L(H, \mathcal{D}_F)$ can be minimized by minimizing $L_k(H, \mathcal{D}_F)$ for $1 \leq k \leq K$ separately. Let $Q_k = \sum_{i=1}^N c_{ik}$. We normalize the costs associated with the k -th part by Q_k to obtain $\mathbf{D}_k = (c_{1k}/Q_k, \dots, c_{Nk}/Q_k)$. $\mathbf{D}_k$ can be considered as a distribution over the training examples of the k -th part in $\mathcal{D}_F$. Boosting can be applied to learn and combine T decision trees to form a detector for the k -th part with example weights initialized to $\mathbf{D}_k$. This learning approach has two limitations: (1) Correlations among the parts are ignored; (2) The computational costs of training and testing increase linearly with the number of parts.

To address the limitations of the separate learning approach, we propose another approach to learn the K part detectors jointly. Instead of learning T decision trees for each part detector, we only learn T decision trees which are shared by all the part detectors. We adapt AdaBoost.MH [147], which is a multi-class, multi-label version of AdaBoost, to learn H of the form:

$$H(x, p) = \sum_{t=1}^T \alpha_t h_t(x, p), \quad (6.5)$$

where $h_t : \mathcal{X} \times \mathcal{P} \rightarrow \mathbb{R}$ is a weak classifier which is a decision tree in our case and α_t is a weight associated with h_t . First, we consider a simplified case in which $c_{ik} = 1$ for $1 \leq i \leq N$ and $1 \leq k \leq K$. AdaBoost.MH can be directly applied to minimize $L(H, \mathcal{D}_F)$. The idea of AdaBoost.MH is to reduce the multi-label learning problem to a binary classification problem for which AdaBoost can be used to obtain H . Each training example $(x_i, \mathbf{Y}_i, \mathbf{C}_i) \in \mathcal{D}_F$ is replaced with K training examples $((x_i, p_k), y_{ik})$ for $1 \leq k \leq K$. Note that since $c_{ik} = 1$ for all i and k , $\mathbf{C}_i$ in the example $(x_i, \mathbf{Y}_i, \mathbf{C}_i)$ can be ignored. y_{ik} is the binary label for (x_i, p_k) . $\mathcal{D}_B = \{((x_i, p_k), y_{ik}) | 1 \leq i \leq N, 1 \leq k \leq K\}$ forms the training set for the binary classification problem. H is learned by minimizing the following loss function:

$$L(H, \mathcal{D}_B) = \frac{1}{N} \sum_{i=1}^N \sum_{k=1}^K \frac{1}{K} [\text{sign}(H(x_i, p_k)) \neq y_{ik}]. \quad (6.6)$$

AdaBoost.MH maintains a sequence of weight matrices $(\mathbf{W}^1, \dots, \mathbf{W}^T)$ through T stages where $\mathbf{W}^t = (w_{ik}^t) \in \mathbb{R}^{N \times K}$ for $1 \leq t \leq T$ with w_{ik}^t the weight of the training example (x_i, p_k) at stage t . $\mathbf{W}^1$ is initialized to $w_{ik}^1 = \frac{1}{NK}$ for all i and k . At each stage t , AdaBoost.MH learns a weak classifier h_t and a weight α_t based on $\mathbf{W}^t$. With h_t , example weights are updated as follows:

$$w_{ik}^{t+1} = \frac{w_{ik}^t \exp(-\alpha_t y_{ik} h_t(x_i, p_k))}{Z_t}, \quad (6.7)$$

where $Z_t = \sum_{i,k} w_{ik}^t \exp(-\alpha_t y_{ik} h_t(x_i, p_k))$ is a normalization factor. The training error of the learned H is bounded by

$$L(H, \mathcal{D}_F) \leq \prod_{t=1}^T Z_t. \quad (6.8)$$

Now we introduce how to minimize $L(H, \mathcal{D}_F)$ for the general case. Note that when $\frac{1}{K}$ in Eq. (6.6) is replaced with c_{ik} , the loss function in Eq. (6.6) is exactly the loss function in Eq. (6.2). It is easy to verify that by initializing $\mathbf{W}^1$ to $w_{ik}^1 = \frac{c_{ik}}{N}$ for all i and k , AdaBoost.MH can be used to minimize $L(H, \mathcal{D}_F)$ in Eq. (6.2). The upper loss bound given in Eq. (6.8) still holds.

Next, we describe how to learn a decision tree h_t at stage t given example weights $\mathbf{W}^t$. Starting with a root node, we construct h_t greedily. At the beginning, all training examples fall in the root node with sample weights $\mathbf{W}^t$. We examine one leaf node at a time. If a leaf node reaches a predefined maximum depth or the training examples falling in the node are pure enough, we stop branching the leaf node. Otherwise, we choose a feature and a threshold which have the minimum weighted classification error on the training examples at the leaf node. With the chosen feature and threshold, the training examples are split into two subsets each of which is assigned to one new leaf node. The two leaf nodes are the children of the current leaf node. Assume h_t has M leaf nodes and the instance space $\mathcal{X}$ is partitioned into $X_1, \dots, X_M$ with X_j the set of instances falling in the j -th leaf node. For an instance $x \in X_j$, h_t is defined to output

$$h_t(x, p_k) = \frac{1}{2} \ln \left(\frac{S_{jk}^+}{S_{jk}^-} \right), \quad (6.9)$$

where $S_{jk}^+ = \sum_{x_i \in X_j} w_{ik}^t [y_{ik} = 1]$ and $S_{jk}^- = \sum_{x_i \in X_j} w_{ik}^t [y_{ik} = -1]$. After the decision tree is constructed, it can be proved that h_t defined in Eq. (6.9) minimizes Z_t with $\alpha_t = 1$ (See [147] for more details).

According to the above adaptation of AdaBoost.MH for minimizing $L(H, \mathcal{D}_F)$, the costs $\mathbf{C} = (c_{ik}) \in \mathbb{R}^{N \times K}$ after normalization can be considered as a distribution over $\mathcal{D}_B$. The decision trees are learned to capture the overall distribution of all the parts. Part correlations are exploited by sharing the decision trees among these parts. When taken as a pedestrian detector individually, the part detectors learned jointly show better performance than those learned separately as demonstrated in Section 6.4. For detection, applying the part detectors with shared decision trees is much faster as it only involves a computational cost of K instead of $K \times T$

decision trees.

6.2.4 Complexity analysis

In general, when part detectors are learned separately, the computational complexity of applying these part detectors for detection is $O(K \times C)$, where K and C are the number of part detectors and the cost of applying each part detector respectively. In our approach, T decision trees are shared by all the K part detectors and each leaf in a decision tree stores K responses. Assume the maximum depth of each decision tree is D . The computation cost of applying one decision tree is $D - 1$ comparisons + K additions. So, the total computational complexity is $T \times (D - 1 + K)$ operations or $O(T \times (D + K))$. When the part detectors are learned separately, each part detector consists of T decision trees. The computational cost of applying one part detector is $O(T \times D)$. So, the total computational complexity is $O(K \times T \times D)$. Compared with separate learning, our approach achieves a speedup of $O((K \times D)/(D + K))$.

6.3 Occlusion handling with part detectors

6.3.1 Integrating part detectors

In a particular scene, pedestrians may be occluded by each other or other objects. Simply applying a full body detector usually does not work well when pedestrians are heavily occluded. As we do not know in advance which parts are occluded, a simple yet effective way to handle occlusions is to apply a set of part detectors. For a candidate region x in an image, the K part detectors would give K detection scores, $\mathbf{s} = (s_1, \dots, s_K)$, where $s_k = H(x, p_k)$ for $1 \leq k \leq K$. We need to integrate these detection scores properly to give a final score indicating how likely the candidate region contains a pedestrian. We study three methods for this purpose.

FIGURE 6.4. Part scores on top 5 scoring image regions. The number above each region denotes its ranking. For the fully visible pedestrian (Red bounding box), all part detectors give high detection scores consistently (Red curve). For the partially occluded pedestrian (Blue bounding box), only the detectors of visible parts (e.g. P3, P4, P5, P9, P10, P14 and P15) output high detection scores (Blue curve). Background regions (Green, yellow and cyan bounding boxes) receive relatively low scores from all the part detectors (Green, yellow and cyan curves).

Let $g : \mathbb{R}^K \rightarrow \mathbb{R}$ be an integration function. A common integration method is to take the maximum among the K detection scores:

$$g(\mathbf{s}) = \max_{1 \leq k \leq K} s_k. \quad (6.10)$$

We call this method Max integration.

The second method we adopt is a variant of Max integration which takes the average of the top S detection scores with $1 \leq S \leq K$. Let $\mathbf{s}'$ be a vector obtained by sorting the detection scores in $\mathbf{s}$ in descending order. The integration function can be expressed as

$$g(\mathbf{s}) = \frac{1}{S} \sum_{1 \leq k \leq S} s'_k. \quad (6.11)$$

We call this method TopS integration. This method is based on two observations as illustrated in Fig. 6.4: (1) for a partially occluded pedestrian, detectors of those parts which are inside or have a large overlap with the visible region of the pedestrian would probably give high detection scores, while the other part detectors may give low detection scores due to occlusions; (2) for a non-pedestrian region, part detectors tend to output low detection scores.

The third method learns a weight vector $\mathbf{v} = (v_1, \dots, v_K)$ to linearly combine the K detection scores:

$$g(\mathbf{s}) = \sum_{1 \leq k \leq K} v_k s_k + b. \quad (6.12)$$

where b is a bias term. Let $\mathcal{D} = \{(\mathbf{s}_i, y_i) | 1 \leq i \leq N\}$ be a set of training examples, where $\mathbf{s}_i$ represents the K detection scores of the i -th training example and $y_i \in \{-1, 1\}$. We learn $\mathbf{w}$ by logistic regression. Specifically, $\mathbf{w}$ is learned by solving the following optimization problem:

$$\min_{\mathbf{v}} \phi(\mathbf{v}) + \lambda \sum_{i=1}^N \log(1 + \exp(-y_i g(\mathbf{s}_i))), \quad (6.13)$$

where $\phi(\mathbf{v})$ is a regularization term. As in Chapter 5, we adopt two types of regularizers, i.e. $\phi(\mathbf{v}) = \frac{1}{2} \mathbf{v}^T \mathbf{v}$ and $\phi(\mathbf{v}) = \|\mathbf{v}\|_1$. When the L1 norm $\|\mathbf{v}\|_1$ is used, a subset of part detectors are selected with a proper choice of λ . We call this method L1 integration if the L1 norm is used in Eq. (6.13) and otherwise L2 integration.

6.3.2 Contextual rescoring

Context is commonly used to improve performance for object detection [10, 119, 148]. We exploit context based on the observation that for our part detectors, detections are densely distributed around a true detection while rare detections are found around a false detection as illustrated in Fig. 6.5. For each detection d , we collect M top-scoring neighboring detections whose intersection over union (IOU) with d is greater than 0.5. Let q be the score for the detection d and q_i^{nb} be the scores for the i -th neighboring detection. These scores come from a specific integration method, e.g. L2 integration. The scores q_i^{nb} ($1 \leq i \leq M$) are sorted in descending order. We represent the detection d by a score vector $\mathbf{u} = (q, q_1^{\text{nb}}, \dots, q_M^{\text{nb}})$ with $q_1^{\text{nb}} \geq \dots \geq q_M^{\text{nb}}$. If the detection d only has less than M neighboring detections, the remaining entries in $\mathbf{u}$ are filled with 0. The score vector of a true detection tends to have more high scores than the score vector of a false detection, which can be exploited to further distinguish between true and false detections. To do this, we learn a rescoring model based on the above

FIGURE 6.5. Detection distribution. Red bounding boxes are thirty detections with the highest scores. Detections are densely located around the two pedestrians marked by green bounding boxes. Detections are sparsely distributed in other regions.

representation using a non-linear support vector machine (SVM) with an intersection kernel. Specifically, we learn a rescoring function of the following form:

$$r(\mathbf{u}) = \sum_{i=1}^N \alpha_i \kappa(\mathbf{u}, \mathbf{u}_i), \quad (6.14)$$

where N is the number of training examples, $\mathbf{u}_i$ is the i -th training example, α_i for $1 \leq i \leq N$ are the parameters to be learned and

$$\kappa(\mathbf{u}, \mathbf{u}_i) = \sum_{j=1}^{M+1} \min(\mathbf{u}(j), \mathbf{u}_i(j)), \quad (6.15)$$

with $\mathbf{u}(j)$ the j -th entry in $\mathbf{u}$. To accelerate training and testing, we approximate the intersection kernel by a linear kernel

$$\kappa(\mathbf{u}, \mathbf{u}_i) \approx \mathbf{u}^T \mathbf{u}'_i, \quad (6.16)$$

where u' and u'_i are approximate explicit mappings of u and u_i respectively obtained by the technique proposed in [149]. Then, we learn the rescoring model by a linear SVM.

6.4 Experiments

We apply our approach to two types of features, hand-crafted channel features and features learned by a convolutional neural network (CNN features), and evaluate its effectiveness on two pedestrian detection datasets, Caltech [3] and CUHK [35].

The Caltech dataset is commonly used for evaluating pedestrian detection approaches and provides both visible portion and full body annotations. Following the standard evaluation protocol, we use video sets S0-S5 for training and video sets S6-S10 for testing. A log-average miss rate, which is calculated by averaging miss rates at 9 false positive per-image (FPPI) points sampled evenly in the log-space ranging from 10^{-2} to 10^0 , is used to summarize the detection performance. As the purpose of our approach is to handle occlusions, we evaluate it on three subsets: *Reasonable*, *Partial* and *Heavy*. In the *Reasonable* subset, only pedestrians with at least 50 pixels tall and under no or partial occlusion are used for evaluation. This subset is widely used for evaluating pedestrian detection approaches. In the *Partial* and *Heavy* subsets, pedestrians are at least 50 pixels tall and are partially occluded (1-35 percent occluded) and heavily occluded (36-80 percent occluded) respectively.

The CUHK dataset is specially collected for evaluating occlusion handling approaches for pedestrian detection. The dataset consists of 1063 images from Caltech, ETHZ, TUD-Brussels, INRIA, Caviar and other sources. In this dataset, each image contains at least one partially occluded pedestrian. As this dataset only contains a small number of images, we do not train detectors on this dataset. Instead, we use this dataset to evaluate the generalizability of detectors trained on the Caltech dataset. Similar to the Caltech dataset, evaluation is done on the *Reasonable*, *Partial* and *Heavy* subsets, and detection performance is summarized by the log-average miss rate.

6.4.1 Experiments with channel features

6.4.1.1 Implementation

We choose locally decorrelated channel features (LDCF) [107] which are frequently used for pedestrian detection in recent years to represent the parts in our approach. We use the same setting as in [107]: 4 filters of size 5×5 are learned to locally decorrelate aggregated channel features (ACF) [11] of 10 channels to generate LDCF of 40 channels. We sample training data from video sets S0-S5 at an interval of 3 frames. Pedestrian examples which are at least 50 pixels tall and occluded not more than 70% are collected as positive examples. Five rounds of bootstrapping are adopted to train 64, 512, 1024, 2048 and 4096 decision trees respectively. The maximum depth of a decision tree is 5. We use $S = 15$ for the TopS integration method. $C = 0.1$ and $C = 0.001$ are used for L2 integration and L1 integration respectively. We use LIBLINEAR [142] to learn model parameters for L1 integration, L2 integration and contextual rescoreing.

6.4.1.2 Experimental results on Caltech

Figure 6.6 shows the results of part detectors learned using different part representations. PR1 denotes the representation method in which a part is represented by the features from its own image region. The part detectors using PR1 are learned independently. PR2 is the representation method in which all the parts share the features from the whole body. The part detectors using PR2 are learned jointly using our multi-label formulation. It can be seen that the performances of the part detectors learned using PR1 vary largely from part to part on the *Reasonable*, *Partial* and *Heavy* subsets. The detectors of small parts (e.g. P5, P10 and P20) usually perform worse than those of large parts (e.g. P1, P6 and P11) since with PR1, the information from the small parts is relatively limited compared with that from the large parts (See Fig. 6.2 for part correspondence). The part detectors using PR2 perform much better than those using PR1. The performances of different part detectors using PR2 do not change much on the three subsets. Although these part detectors show similar performances, they do behave differently.

FIGURE 6.6. Results of part detectors using different part representations on the Caltech dataset.

The example distribution of each part is captured by its detector. When a pedestrian example is occluded, those parts inside or have large overlap with the visible portion usually get large detection scores while the other parts tend to have low detection scores (See the blue curve in Fig. 6.4).

Table 6.1 shows the results of the detectors of six typical parts (P1-P4, P6 and P11) learned by two different approaches, separate learning (SL) and joint learning (JL). SL learns part detectors separately by minimizing Eq. (6.3), while JL learns all part detectors jointly by minimizing Eq. (6.2) using AdaBoost.MH. For the six parts, the detectors learned by JL perform better than their counterparts learned by SL on all the three subsets, which shows the effectiveness of sharing decision trees to exploit correlations among the parts. The average improvements on

Part detector	<i>Reasonable</i>	<i>Partial</i>	<i>Heavy</i>
SI-P1	18.2	36.1	72.1
JL-P1	17.0	34.2	70.5
SL-P2	19.3	40.2	71.9
JL-P2	17.4	35.8	70.1
SL-P3	18.6	39.4	69.4
JL-P3	17.2	34.3	68.7
SL-P4	18.6	39.7	69.9
JL-P4	17.8	35.5	67.9
SL-P6	19.2	37.7	72.1
JL-P6	17.3	34.1	70.7
SL-P11	19.2	42.4	73.8
JL-P11	16.9	35.2	70.8
Avg. Imp.	+1.6	+4.4	+1.8

TABLE 6.1. Comparison of separate learning (SL) and joint learning (JL) using channel features on the Caltech dataset. P1-P4, P6 and P11 are six typical parts shown in Fig. 6.2. The last row shows the average improvements on the three subsets brought by joint learning.

part detector	<i>Reasonable</i>	<i>Partial</i>	<i>Heavy</i>
HI-P1	20.9	44.2	79.5
JL-P1	17.0	34.2	70.5
HL-P2	22.1	43.7	77.9
JL-P2	17.4	35.8	70.1
HL-P3	24.5	46.0	74.1
JL-P3	17.2	34.3	68.7
HL-P4	31.4	53.2	73.6
JL-P4	17.8	35.5	67.9
HL-P6	22.6	44.0	82.0
JL-P6	17.3	34.1	70.7
HL-P11	22.4	46.3	83.0
JL-P11	16.9	35.2	70.8
Avg. Imp.	+6.7	+11.4	+8.6

TABLE 6.2. Comparison of cost-sensitive labeling and hard learning using channel features on the Caltech dataset. P1-P4, P6 and P11 are six typical parts shown in Figure. 6.2. The last row shows the average improvements on the three subsets brought by cost-sensitive labeling.

Subset	Full-body detectors				Integration methods			
	LDCF-P1	SL-P1	HL-P1	JL-P1	JL-Max	JL-TopS	JL-L1	JL-L2
<i>Reasonable</i>	18.1	18.2	20.9	17.0	17.5	16.6	16.7	16.5
<i>Partial</i>	38.8	36.1	44.2	34.2	34.8	32.7	33.1	33.0
<i>Heavy</i>	72.4	72.1	79.5	70.5	68.8	69.6	69.0	68.9

TABLE 6.3. Results of full-body detectors and integration approaches using channel features on the Caltech dataset.

the three subsets brought by JL are 1.6%, 4.4% , and 1.8% respectively.

Table 6.2 compares two labeling strategies: cost-sensitive labeling and hard labeling. The proposed joint learning approach adopts a cost-sensitive labeling strategy: besides a class label, each training example is also associated with a misclassification cost (See Section 6.2.2). To demonstrate the effectiveness of the cost-sensitive labeling strategy, we consider a hard labeling strategy in which every training example has a misclassification cost of 1. JL is our approach with cost-sensitive labeling and HL also learns the part detectors jointly by minimizing Eq. (6.2) but with the hard labeling strategy. The detectors of six typical parts (P1-P4, P6 and P11) learned by JL perform significantly better than their counterparts learned by HL with mean improvements of 6.7%, 11.4% and 8.6% on the *Reasonable*, *Partial* and *Heavy* subsets respectively. From the comparison, we can see that proper labeling of training examples is important for our multi-label learning approach to work well.

Table 6.3 shows the results of different full-body detectors and integration methods. LDCF-P1 is our implementation of LDCF [107] which only uses fully visible pedestrian examples as positive examples. LDCF-P1 and SL-P1 performs closely on the *Reasonable* and *Heavy* subsets, but SL-P1 outperforms LDCF-P1 by 2.7% on the *Partial* subset since SL-P1 uses additional partially occluded pedestrian examples for training according to the definition of the misclassification cost in Eq. (6.1). JL-P1 achieves the best performance among the four full-body detectors on all the three subsets. Particularly, JL-P1 outperforms SL-P1 by 1.2%, 1.9% and 1.6% on the three subsets respectively. JL-Max, JL-TopS, JL-L1 and JL-L2 represent the four integration methods (Max, TopS, L1 and L2 described in Section 6.3.1) applied to the 20 part detectors learned by JL. Except JL-Max, the other three integration methods outperform

part pool	method	<i>Reasonable</i>	<i>Partial</i>	<i>Heavy</i>
PP5	JL-P1	17.7	34.3	70.7
	JL-L2	18.0	35.0	68.5
	JL-L2-Ctx	17.1	34.4	67.2
PP10	JL-P1	17.5	34.6	70.2
	JL-L2	17.7	34.5	70.4
	JL-L2-Ctx	17.3	33.7	69.3
PP20	JL-P1	17.0	34.2	70.5
	JL-L2	16.5	33.0	68.9
	JL-L2-Ctx	15.9	31.0	68.3

TABLE 6.4. Results of approaches using different part pools on the Caltech dataset.

JL-P1 on all the three subsets, which demonstrates that properly integrating part detectors can improve the performance in different occlusion situations. JL-TopS performs slightly worse than JL-Max on the *Heavy* subset, but outperforms it by 0.9% and 2.1% on the *Reasonable* and *Partial* subsets respectively. Overall, TopS integration is more robust than Max integration. JL-TopS, JL-L1 and JL-L2 perform closely on the three subsets. Particularly, JL-L1 does not show advantage over JL-L2, which is different from the results reported in Chapter 5. This is because the part detectors used in Chapter 5 are learned independently using PR1 while JL-L1 and JL-L2 uses the part detectors learned jointly using PR2. The performances of the detectors learned using PR1 vary largely while the part detectors learned jointly using PR2 are more robust as illustrated in Fig. 6.6. In Chapter 5, the L1 regularization can remove unstable detectors to eliminate the negative effects of these detectors. For JL-L1 and JL-L2, all the part detectors show stable performances and it is not necessary to remove any part detectors. Contrarily, JL-L2 shows slightly better performance than JL-L1 on the three subsets. In addition, the advantage of JL-L2 over JL-Max is not as significant as that reported in Chapter 5. This is because the part detectors used in JL-L2 are learned jointly, which makes their outputs more comparable.

Table 6.4 shows the results of three approaches (JL-P1, JL-L2 and JL-L2-Ctx) using part pools PP5, PP10 and PP20. JL-L2-Ctx is an approach which combines JL-L2 and the contextual rescoring model described in Section 6.3.2. PP20 is the part pool defined in Fig. 6.2. PP5 and PP10 are two subsets of PP20 which are shown in Fig. 6.7(a) and Fig. 6.7(b) respectively. For

(a) PP5

(b) PP10

FIGURE 6.7. Two part pools.

all the three part pools, JL-L2 performs better than JL-P1 and JL-L2-Ctx performs best on the three subsets, which shows that both L2 integration and contextual rescoring contribute to the performance improvement. Among the three part pools, the three approaches work best with PP20 since a larger part pool can better cover occlusion situations in the dataset.

Figure 6.8 gives a comparison of our approach and state-of-the-art approaches using channel features, ACF [11], InformedHaar [105], NAMC [150], LDCF [107], Katamari [100], SpatialPooling+ [106], SCCPriors [151] and Checkerboards [108]. Our approach achieves the best performance among these channel-feature based approaches. Our full-body detector (JL-P1)

FIGURE 6.8. Comparison with state-of-the-art approaches using channel features on the Caltech dataset.

already outperforms Checkerboards on all the three subsets. The improvement of JL-P1 over Checkerboards is significant (7%) on the *Heavy* subset. With L2 integration and contextual rescoring, JL-L2-Ctx outperforms JL-P1 on the three subsets by 1.1%, 3.2% and 2.2% respectively. Both JL-P1 and JL-L2-Ctx outperform Checkerboards significantly (7% and 9.2% respectively) on the *Heavy* subset, which demonstrates the effectiveness of our approach for detecting heavily occluded pedestrians.

6.4.1.3 Experimental results on CUHK

We compare our approach with Checkerboards [108] on this dataset. Figure 6.9 shows the results of Checkerboards, JL-P1 and JL-L2-Ctx. For Checkerboards, we use the code and its

FIGURE 6.9. Results using channel features on the CUHK dataset.

model trained on the Caltech dataset which are publicly available to generate the result. JL-P1 performs 2% worse than Checkerboards on the *Reasonable* subset but outperforms Checkerboards by 7.9% and 9.3% on the *Partial* and *Heavy* subsets respectively, which shows the advantage of learning the full-body detector jointly with other part detectors to exploit part correlations. With L2 integration and contextual rescoring, JL-L2-Ctx outperforms JL-P1 by 11.8%, 16.6% and 19.7% on the *Reasonable*, *Partial* and *Heavy* subsets respectively. JL-L2-Ctx shows significant advantage over Checkerboards on all the three subsets.

6.4.2 Experiments with CNN features

6.4.2.1 Implementation

Recently, several approaches using CNN features have achieved the state-of-the-art performance for pedestrian detection [32, 92, 113, 114]. The proposed multi-label learning approach also applies to CNN features. We use a region proposal network (RPN) from [114] for feature extraction and then learn a set of part detectors jointly as described in Section 6.2.3. RPN+BF [114] also adopts a similar framework in which a set of decision trees are learned to form a full-body detector using CNN features from the RPN. The major differences between RPN+BF and our approach are two-fold: (1) our approach jointly learns the full-body detector with the other part detectors to exploit part correlations; (2) our approach further integrates the part detectors to better handle occlusions. We sample training data from video sets S0-S5 at an interval of 3 frames as in [114]. Pedestrian examples which are at least 50 pixels tall and occluded not more than 70% are collected as positive examples. These positive examples are also used for training the RPN (See [114] for the network architecture and training procedure of the PRN). To speed up training and testing, we use the RPN to generate pedestrian proposals. About 1000 proposals and 400 proposals per image are generated for training and testing respectively. Six rounds of bootstrapping are adopted to train 64, 128, 256, 512, 1024 and 2048 decision trees respectively. The maximum depth of a decision tree is 5. We use $S = 15$ for the TopS integration method. $C = 0.1$ and $C = 0.005$ are used for L2 integration and L1 integration respectively. We use LibLinear to learn model parameters for L1 integration, L2 integration and contextual rescoreing.

6.4.2.2 Detection speed

On an NVIDIA K5200 GPU, it takes about 0.6s (0.5s for feature extraction and 0.1s for detection) to test the jointly learned part detectors on a 480×640 image, while it takes about 2.2s (0.5s + 1.7s) to apply 20 separately learned detectors. Excluding the time for feature extraction, the speedup factor of the jointly learned part detectors is close to $20\times$.

6.4.2.3 Experimental results on Caltech

Table 6.5 shows the results of different full-body detectors and integration methods using CNN features. RPN+BF-P1 is our implementation of RPN+BF [114]. SL-P1 and JL-P1 are two full-body detectors learned by separate learning (SL) and joint learning (JL) respectively. SL-P1 outperforms slightly worse than RPN+BF-P1 on the *Reasonable* subset but outperforms it on the *Partial* and *Heavy* subsets. The use of some partially occluded pedestrian examples for training makes SL-P1 achieve better performance for occluded pedestrian detection. JL-P1 outperforms SL-P1 on the three subsets by 0.4% (*Reasonable*), 0.8% (*Partial*) and 6.1% (*Heavy*) respectively. The performance improvement on *Heavy* is significant. In our multi-label learning approach, the full-body detector (JL-P1) is learned jointly with the other part detectors by sharing decision trees. These decision trees are learned to capture the overall distribution of pedestrian examples including heavily occluded ones. When the full-body detector is learned separately, most heavily occluded pedestrian examples are ignored, which makes SL-P1 perform relatively poorly on the *Heavy* subset. Using CNN features, HL-P1 performs much worse than JL-P1, which is consistent with the case of channel features. JL-Max, JL-TopS, JL-L1 and JL-L2 are four methods which use Max, TopS, L1 and L2 respectively to integrate the 20 part detectors learned by JL. The four integration methods show better performance than the single full-body detector JL-P1 on the *Partial* and *Heavy* subsets, which justifies that properly integrating part detectors can better handle occlusions than a single full-body detector. On the *Reasonable* subset, all the four integration methods perform slightly worse than JL-P1. Since JL-P1 already works well for detecting pedestrians which are non-occluded or slightly occluded, integrating the other part detectors with the full-body detector does not help. JL-Max has better performance than JL-TopS on the *Heavy* subset, while JL-TopS outperforms JL-Max on the *Reasonable* and *Partial* subsets. JL-TopS, JL-L1 and JL-L2 have similar performances.

Figure 6.10 gives a comparison of our approach and some state-of-the-art CNN-feature based approaches, TA-CNN [112], CCF [111], CCF+CF [111], DeepParts [32], CompACT-Deep [113], MS-CNN [92] and RPN+BF [114]. On the *Reasonable* subset, JL-P1 performs comparably to the top two approaches RPN+BF and MS-CNN which also only use a single full-body

Subset	Full-body detectors				Integration methods			
	RPN+BF-P1	SL-P1	HL-P1	JL-P1	JL-Max	JL-TopS	JL-L1	JL-L2
<i>Reasonable</i>	10.1	10.3	10.8	9.9	10.3	10.0	10.0	10.0
<i>Partial</i>	18.9	18.0	22.9	17.2	17.2	16.6	16.5	16.6
<i>Heavy</i>	58.9	56.6	66.6	50.5	48.4	49.2	49.3	49.2

TABLE 6.5. Results of full-body detectors and integration approaches using CNN features on the Caltech dataset.

FIGURE 6.10. Comparison with state-of-the-art approaches using CNN features on the Caltech dataset.

FIGURE 6.11. Comparison with new state-of-the-art approaches on the Caltech dataset.

detector. This is because the three approaches use similar deep convolutional neural networks (variants of VGG-16 [152]). On the *Partial* and *heavy* subsets, JL-P1 outperforms the most competitive approach MS-CNN by 2.0% and 9.4% respectively. The advantage of JL-P1 over MS-CNN on *Heavy* is significant, which shows the effectiveness of learning the full-body detector jointly with the other part detectors. With L2 integration and contextual rescoring, JL-L2-Ctx further improves the performance over JL-P1, especially on the *Partial* and *Heavy* subsets. JL-L2-Ctx outperforms MS-CNN by 0.2%, 3.6% and 11.3% on the *Reasonable*, *Partial* and *Heavy* subsets respectively. We also compare our approach with some newly published approaches, MS-RCNN [92], RPN+BF [114], F-DNN [144], F-DNN+SS [144], DeepParts [32], CompACT-Deep [113], SDS-RCNN [145] and PCN [146] in Fig. 6.11. Figure 6.12 shows some detection examples of our approach on the Caltech dataset.

FIGURE 6.12. Detection examples of our approach on the Caltech dataset.

6.4.2.4 Experimental results on CUHK

We compare our approach with RPN+BF [114] and MS-CNN [92] on this dataset. For RPN+BF and MS-CNN, we use the published models trained on the Caltech dataset. Figure 6.13 shows the results of RPN+BF, MS-CNN, JL-P1 and JL-L2-Ctx. MS-CNN performs poorly on this dataset. JL-P1 performs 2.9% worse than RPN+BF on the *Reasonable* subset, but outperforms it by 7% and 27% on the *Partial* and *Heavy* subsets, which demonstrates the effectiveness of our multi-label learning approach. With linear integration and contextual rescoring, JL-L2-Ctx achieves better performance than JL-P1. Figure 6.14 shows some detection examples of our

FIGURE 6.13. Results on CUHK using CNN features.

approach on the CUHK dataset.

6.5 Conclusion

In this chapter, we propose a multi-label learning approach to learn part detectors jointly for the framework of learning a part detector ensemble to handle occlusions in pedestrian detection. AdaBoost.MH is adapted to learn a set of decision trees which are shared by all the part detectors. Thanks to the sharing of decision trees, part correlations are exploited and the computational cost of applying these part detectors is reduced. The learned decision trees capture the overall distribution of all the parts. We explore several methods to integrate the part

FIGURE 6.14. Detection examples of our approach on the CUHK dataset.

detectors learned by the proposed approach for occlusion handling. We also propose a contextual rescoring model to further improve the performance. The proposed approach is applied to hand-crafted channel features and CNN features respectively. Its effectiveness is validated on the Caltech and CUHK datasets. The experimental results show that (1) the part detectors learned jointly by the proposed approach perform better than their counterparts learned separately; (2) proper integration of these part detectors can improve the performance for detecting occluded pedestrians, especially heavily occluded ones; (3) contextual rescoring can further improve the performance for pedestrian detection.

Chapter 7

Conclusion

Object detection is a fundamental problem in computer vision and has many applications such as face detection in digital cameras, video surveillance and autonomous driving. Except for face, object detection has not been well solved for other categories. One of the major challenges comes from large intra-category appearance variations which can be caused by factors including subcategory, viewpoint, deformation and occlusion. This thesis studies how to exploit detector ensembles for deformations and occlusions, particularly the latter situation, for generic object detection as well as pedestrian detection.

In the first part of the thesis, two approaches are proposed to handle deformations and occlusions respectively for generic object detection based on a classic part detector ensemble, deformable part model (DPM). Chapter 3 presents an approach to part discovery for the DPM in which parts play an important role. Instead of a set of rectangle parts obtained in a heuristic way, non-rectangular parts are discovered by taking object structures into account. We formulate part discovery as a segmentation problem which is solved by normalized cuts followed by local refinement. With the non-rectangular parts discovered by the approach, the learned DPMs can better capture part appearance and structural deformations of object categories. Chapter 4 studies how to discover occlusion patterns for generic object categories to handle occlusions. Based on the DPM, an occlusion pattern is defined to consist of a subset of parts and a pool of

an exponential number of occlusion patterns is constructed. Occlusion pattern discovery is formulated as a facility-location problem which is efficiently solved by a greedy search algorithm to select a subset of occlusion patterns that are both representative and discriminative from the occlusion pattern pool. The discovered occlusion patterns are themselves DPMs and can be used as object detectors when properly tuned. They can also be combined with state-of-the-art detectors (e.g. Faster R-CNN) for improving detection performance and achieving part-level occlusion reasoning.

The second part of the thesis focuses on how to handle occlusions for pedestrian detection. Two approaches are proposed to improve the framework of learning an ensemble of part detectors. Chapter 5 studies how to properly integrate part detectors for heavily occluded pedestrian detection. Instead of combining all the part detectors to form a generic classifier for all occlusions, we categorize occlusions into several groups based on how pedestrian examples are occluded. A subset of relevant and reliable part detectors are selected and linearly combined to obtain a classifier for each occlusion group. Using different classifiers for these occlusion groups achieves better performance for heavily occluded pedestrian detection than using a generic classifier which combines all the part detectors. Chapter 6 presents a multi-label learning approach to learn part detectors jointly. The part detectors share a set of decision trees via boosting to exploit part correlations and also reduce the computational cost of applying the part detectors. Thanks to the sharing of decision trees, part correlations are exploited and the computational cost of applying these part detectors is reduced. The part detectors learned jointly by the proposed approach perform better than their counterparts learned separately. Several methods are explored to integrate the part detectors for handling occlusions. Experimental results show that these part detectors are reliable and not sensitive to different integration methods. Context is also exploited to further improve the pedestrian detection performance.

Publications

- Youmei Zhang, Chunluan Zhou, Faliang Chang and Alex Kot, "Attention to Head Locations for Crowd Counting", Submitted to *IEEE Transactions on Multimedia (TMM)*, 2018, **Under review**.
- Chunluan Zhou and Junsong Yuan, "Occlusion Pattern Discovery for Object Detection and Occlusion Reasoning", Submitted to *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*, 2018, **Under review**.
- Chunluan Zhou and Junsong Yuan, "Multi-label Learning of Part Detectors for Occluded Pedestrian Detection", *Pattern Recognition (PR)*, 2018.
- Chunluan Zhou and Junsong Yuan, "Bi-box Regression for Pedestrian Detection and Occlusion Estimation", *European Conference on Computer Vision (ECCV)*, 2018.
- Kang Dang, Chunluan Zhou, Zhigang Tu, Justin Dauwels and Junsong Yuan, "Actor-Action Semantic Segmentation with Region Masks", *British Machine and Vision Conference (BMVC)*, 2018.
- Chunluan Zhou and Junsong Yuan, "Multi-label Learning of Part Detectors for Heavily Occluded Pedestrian Detection", Submitted to *IEEE Conference on Computer Vision and Pattern Recognition (ICCV)*, 2017.
- Chunluan Zhou and Junsong Yuan, "Learning to Integrate Occlusion-specific Detectors for Heavily Occluded Pedestrian Detection", *Asian Conference on Computer Vision (ACCV)*, 2016.

- Chunluan Zhou and Junsong Yuan, "Non-rectangular Part Discovery for Object Detection", *British Machine and Vision Conference (BMVC)*, 2014.
- Chunluan Zhou and Junsong Yuan, "Arbitrary-shape Object Localization using Adaptive Image Grids", *Asian Conference on Computer Vision (ACCV)*, 2012.

Bibliography

- [1] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman, “The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results,” <http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html>, 2007.
- [2] R. Girshick, P. Felzenszwalb, and D. McAllester, “Object detection with grammar models,” in *Advances in Neural Information Processing Systems (NIPS)*, 2011, pp. 442–450.
- [3] P. Dollar, W. C., S. B., and P. P., “Pedestrian detection: An evaluation of the state of the art,” *IEEE Transaction on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 34, no. 4, pp. 743–761, 2012.
- [4] M. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. Van Gool, “Robust tracking-by-detection using a detector confidence particle filter,” in *International Conference on Computer Vision (ICCV)*, 2009, pp. 1515–1522.
- [5] T. Lan, Y. Wang, S. Robinovitch, and G. Mori, “Discriminative latent models for recognizing contextual group activities,” *IEEE Transaction on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 34, no. 8, pp. 1549–1562, 2012.
- [6] A. Karpathy and F. Li, “Deep visual-semantic alignments for generating image descriptions,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3128–3137.

- [7] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001, pp. 511–518.
- [8] R. Girshick, J. Donahue, T. Darrel, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 580–587.
- [9] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman, “Multiple kernels for object detection,” in *International Conference on Computer Vision (ICCV)*, 2009, pp. 606–613.
- [10] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan, “Object detection with discriminatively trained part-based models,” *IEEE Transaction on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 32, no. 9, pp. 1627–1645, 2010.
- [11] P. Dollar, R. Appel, S. Belongie, and P. Perona, “Fast feature pyramids for object detection,” *IEEE Transaction on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 36, no. 8, pp. 1532–1545, 2014.
- [12] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005, pp. 886–893.
- [13] X. Wang, M. Yang, S. Zhu, and Y. Lin, “Regionlets for generic object detection,” in *International Conference on Computer Vision (ICCV)*, 2013, pp. 17–24.
- [14] R. Girshick, “Fast r-cnn,” in *International Conference on Computer Vision (ICCV)*, 2015, pp. 1440–1448.
- [15] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” in *Advances in Neural Information Processing Systems (NIPS)*, 2015, pp. 91–99.
- [16] S. Dai, M. Yang, Y. Wu, and A. Katsaggelos, “Detector ensemble,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007, pp. 1–8.

-
- [17] X. Zhang, H. Xiong, W. Lin, and Q. Tian, “Ensemble of part detectors for simultaneous classification and localization,” *CoRR*, vol. abs/1705.10034, 2017.
- [18] C. Gu, P. Arbelaez, Y. Lin, K. Yu, and J. Malik, “Multi-component models for object detection,” in *European Conference on Computer Vision (ECCV)*, 2012, pp. 445–458.
- [19] M. Hoai and A. Zisserman, “Discriminative sub-categorization,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 1666–1673.
- [20] X. Zhu, D. Anguelov, and D. Ramanan, “Capturing long-tail distributions of object sub-categories,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 915–922.
- [21] H. Bilen, M. Pedersoli, and T. Tuytelaars, “Weakly supervised object detection with convex clustering,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1081–1089.
- [22] H. Azizpour and I. Laptev, “Object detection using strongly-supervised deformable part models,” in *European Conference on Computer Vision (ECCV)*, 2012, pp. 836–849.
- [23] B. Li, W. Hu, T. Wu, and S. Zhu, “Modeling occlusion by discriminative and-or structures,” in *International Conference on Computer Vision (ICCV)*, 2013, pp. 2560–2567.
- [24] B. Pepik, M. Stark, P. Gehler, and B. Schiele, “Occlusion patterns for object class detection,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 3286–3293.
- [25] B. Li, T. Wu, and S. Zhu, “Integrating context and occlusion for car detection by hierarchical and-or model,” in *European Conference on Computer Vision (ECCV)*, 2014, pp. 652–667.
- [26] Y. Li, S. Wang, Q. Tian, and X. Ding, “Learning cascaded shared-boost classifiers for part-based object detection,” *IEEE Transactions on Image Processing (TIP)*, vol. 23, no. 4, pp. 1858–1871, 2014.

- [27] D. Crandall, P. Felzenszwalb, and D. Huttenlocher, “Spatial priors for part-based recognition using statistical models,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005, pp. 10–17.
- [28] R. Fergus, P. Perona, and A. Zisserman, “A sparse object category model for efficient learning and exhaustive recognition,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005, pp. 380–387.
- [29] P. Felzenszwalb and D. Huttenlocher, “Pictorial structures for object recognition,” *International Journal of Computer Vision*, vol. 61, no. 1, pp. 55–79, 2005.
- [30] B. Ommer and J. Buhmann, “Learning compositional categorization models,” in *European Conference on Computer Vision (ECCV)*, 2006, pp. 316–329.
- [31] M. Mathias, R. Benenson, R. Timofte, and L. Van Gool, “Handling occlusions with franken-classifiers,” in *International Conference on Computer Vision (ICCV)*, 2013, pp. 1505–1512.
- [32] Y. Tian, P. Luo, X. Wang, and X. Tang, “Deep learning strong parts for pedestrian detection,” in *International Conference on Computer Vision (ICCV)*, 2015, pp. 1904–1912.
- [33] M. Enzweiler, A. Eigenstetter, B. Schiele, and D. Gavrilu, “Multi-cue pedestrian classification with partial occlusion handling,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010, pp. 990–997.
- [34] G. Duan, H. Ai, and S. Lao, “A structural filter approach to human detection,” in *European Conference on Computer Vision (ECCV)*, 2010, pp. 238–251.
- [35] W. Ouyang and X. Wang, “A discriminative deep model for pedestrian detection with occlusion handling,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012, pp. 3258–3265.
- [36] W. Ouyang, X. Zeng, and X. Wang, “Modeling mutual visibility relationship in pedestrian detection,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 3222–3229.

-
- [37] T. Malisiewicz, A. Gupta, and A. Efros, “Ensemble of exemplar-svms for object detection and beyond,” in *International Conference on Computer Vision (ICCV)*, 2011, pp. 89–96.
- [38] J. Redmon and S. Divvala, “You only look once: Unified, real-time object detection,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.
- [39] B. Drayer and T. Brox, “Training deformable object models for human detection based on alignment and clustering,” in *European Conference on Computer Vision (ECCV)*, 2014, pp. 406–420.
- [40] Y. Chen, L. Zhu, and A. Yuille, “Active mask hierarchies for object detection,” in *European Conference on Computer Vision (ECCV)*, 2010, pp. 43–56.
- [41] J. Zhang, K. Huang, Y. Yu, and T. Tan, “Boosted local structured hog-lbp for object localization,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011, pp. 1393–1400.
- [42] C. Dubout and F. Fleuret, “Exact acceleration of linear object detection,” in *European Conference on Computer Vision (ECCV)*, 2012, pp. 301–311.
- [43] X. Ren and D. Ramanan, “Histograms of sparse codes for object detection,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 3246–3253.
- [44] S. Tang, M. Andriluka, and B. Schiele, “Detection and tracking of occluded people,” in *British Machine Vision Conference (BMVC)*, 2012, pp. 9.1–9.11.
- [45] W. Ouyang and W. X., “Single-pedestrian detection aided by multi-pedestrian detection,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 3198–3205.
- [46] F. Khan, R. Anwer, J. Weijer, A. Bagdanov, M. Vanrell, and A. Lopez, “Color attributes for object detection,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012, pp. 3306–3313.

- [47] S. Fidler, R. Mottaghi, A. Yuille, and R. Urtasun, “Bottom-up segmentation for top-down detection,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 3294–3301.
- [48] S. Divvala, A. Efros, and M. Hebert, “How important are ”deformable parts” in the deformable parts model,” in *European Conference on Computer Vision (ECCV), Workshops and Demonstrations*, 2012, pp. 31–40.
- [49] X. Song, T. Wu, Y. Jia, and S. Zhu, “Discriminatively trained and-or models for object detection,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 3278–3285.
- [50] O. Aghazadeh, H. Azizpour, J. Sullivan, and S. Carlsson, “Mixture component identification and learning for visual recognition,” in *European Conference on Computer Vision (ECCV)*, 2012, pp. 115–128.
- [51] T. Lan, M. Raptis, L. Sigal, and G. Mori, “From subcategory to visual components: A multi-level framework for object detection,” in *International Conference on Computer Vision (ICCV)*, 2013, pp. 369–376.
- [52] X. Chen, R. Mottaghi, X. Liu, S. Fidler, R. Urtasun, and A. Yuille, “Detect what you can: Detecting and representing objects using holistic models and body parts,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 1979–1986.
- [53] G. Chen, Y. Ding, J. Xiao, and T. Han, “Detection evolution with multi-order contextual co-occurrence,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 1798–1805.
- [54] Z. Song, Q. Chen, Z. Huang, Y. Hua, and S. Yan, “Contextualizing object detection and classification,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011, pp. 1585–1592.
- [55] O. Parkhi, A. Vedaldi, C. Jawahar, and A. Zisserman, “The truth about cats and dogs,” in *International Conference on Computer Vision (ICCV)*, 2011, pp. 1427–1434.

-
- [56] R. Mottaghi, “Augmenting deformable part models with irregular-shaped object patches,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012, pp. 3116–3123.
- [57] E. Trulls, S. Tsogkas, I. Kokkinos, A. Sanfeliu, and F. Moreno-Noguer, “Segmentation-aware deformable part models,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 168–175.
- [58] D. Cheng, J. Wang, X. Wei, and Y. Gong, “Training mixture of weighted svm for object detection using em algorithm,” *Neurocomputing*, vol. 149, pp. 473–482, 2015.
- [59] R. Girshick, F. Iandola, T. Darrell, and J. Malik, “Deformable part models are convolutional neural networks,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 437–446.
- [60] B. Pepik, M. Stark, P. Gehler, and B. Schiele, “Teaching 3d geometry to deformable part models,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012, pp. 3362–3369.
- [61] —, “Multi-view and 3d deformable part models,” *IEEE Transaction on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 37, no. 11, pp. 2232–2245, 2015.
- [62] D. Levi, S. Silberstein, and A. Bar-Hillel, “Fast multiple-part based object detection using kd-ferns,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 947–954.
- [63] P. Felzenszwalb, R. Girshick, and D. McAllester, “Cascade object detection with deformable part models,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010, pp. 2241–2248.
- [64] M. Pedersoli, A. Vedaldi, and J. Gonzalez, “A coarse-to-fine approach for fast deformable object detection,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011, pp. 1353–1360.

- [65] J. Yan, Z. Lei, L. Wen, and S. Li, “The fastest deformable part model for object detection,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 2497–2504.
- [66] M. Sadeghi and D. Forsyth, “30hz object detection with dpm v5,” in *European Conference on Computer Vision (ECCV)*, 2014, pp. 65–79.
- [67] T. Dean, M. Ruzon, M. Segal, J. Shlens, S. Vijayanarasimhan, and J. Yagnik, “Fast, accurate detection of 100,000 object classes on a single machine,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 1814–1821.
- [68] H. Song, S. Zickler, T. Althoff, R. Girshick, M. Fritz, C. Geyer, P. Felzenszwalb, and T. Darrell, “Sparselet models for efficient multiclass object detection,” in *European Conference on Computer Vision (ECCV)*, 2012, pp. 802–815.
- [69] I. Kokkinos, “Shufflets: Shared mid-level parts for fast object detection,” in *International Conference on Computer Vision (ICCV)*, 2013, pp. 1393–1400.
- [70] R. Girshick and J. Malik, “Training deformable part models with decorrelated features,” in *International Conference on Computer Vision (ICCV)*, 2013, pp. 3016–3023.
- [71] T. Gao, P. B., and K. D., “A segmentation-aware object detection model with occlusion handling,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011, pp. 1361–1368.
- [72] X. Wang, T. Han, and S. Yan, “An hog-lbp human detector with partial occlusion handling,” in *International Conference on Computer Vision (ICCV)*, 2009, pp. 32–39.
- [73] A. Vedaldi and A. Zisserman, “Structured output regression for detection with partial truncation,” in *Advances in Neural Information Processing Systems (NIPS)*, 2009, pp. 1928–1936.
- [74] Y. Xiang, W. Choi, Y. Lin, and S. Savarese, “Data-driven 3d voxel patterns for object category recognition,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1903–1911.

-
- [75] L. Bourdev and J. Malik, “Poselets: Body part detectors trained using 3d human pose annotations,” in *International Conference on Computer Vision (ICCV)*, 2009, pp. 1365–1372.
- [76] L. Bourdev, S. Maji, T. Brox, and J. Malik, “Detecting people using mutually consistent poselet activations,” in *European Conference on Computer Vision (ECCV)*, 2010, pp. 168–181.
- [77] B. Leibe, A. Leonardis, and B. Schiele, “Combined object categorization and segmentation with an implicit shape model,” in *ECCV Workshop on Statistical Learning in Computer Vision*, 2004.
- [78] S. Maji and J. Malik, “Object detection using a max-margin hough transform,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009, pp. 1038–1045.
- [79] W. Ouyang, X. Wang, X. Zeng, S. Qiu, P. Luo, Y. Tian, H. Li, S. Yang, Z. Wang, C. Loy, and X. Tang, “Deepid-net: Deformable deep convolutional neural networks for object detection,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 2403–2412.
- [80] S. Gidaris and N. Komodakis, “Object detection via a multi-region and semantic-aware cnn model,” in *International Conference on Computer Vision (ICCV)*, 2015, pp. 1134–1142.
- [81] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.
- [82] X. Zeng, W. Ouyang, B. Yang, J. Yan, and X. Wang, “Gated bi-directional cnn for object detection,” in *European Conference on Computer Vision (ECCV)*, 2016, pp. 354–369.
- [83] L. Zitnick and P. Dollar, “Edge boxes: Locating object proposals from edges,” in *European Conference on Computer Vision (ECCV)*, 2014, pp. 391–405.

- [84] P. Krahenbuhl and V. Koltun, “Geodesic object proposals,” in *European Conference on Computer Vision (ECCV)*, 2014, pp. 725–739.
- [85] S. He and R. Lau, “Oriented object proposals,” in *International Conference on Computer Vision (ICCV)*, 2015, pp. 280–288.
- [86] Y. Lu and T. Javidi, “Adaptive object detection using adjacency and zoom prediction,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2351–2359.
- [87] Y. Zhang, K. Sohn, R. Villegas, G. Pan, and H. Lee, “Improving object detection with deep convolutional networks via bayesian optimization and structured prediction,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 249–258.
- [88] S. Liu, C. Lu, and J. Jia, “Box aggregation for proposal decimation: Last mile of object detection,” in *International Conference on Computer Vision (ICCV)*, 2015, pp. 2569–2577.
- [89] C. Lu, Y. Lu, H. Chen, and C. Tang, “Square localization for efficient and accurate object detection,” in *International Conference on Computer Vision (ICCV)*, 2015, pp. 2560–2568.
- [90] T. Kong, A. Yao, Y. Chen, and F. Sun, “Hypernet: Towards accurate region proposal generation and joint object detection,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 845–853.
- [91] S. Gidaris and N. Komodakis, “Locnet: Improving localization accuracy for object detection,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 789–798.
- [92] Z. Cai, Q. Fan, R. Feris, and N. Vasconcelos, “A unified multi-scale deep convolutional neural network for fast object detection,” in *European Conference on Computer Vision (ECCV)*, 2016, pp. 354–370.

-
- [93] T. Lin, P. Dollar, and R. Girshick, “Feature pyramid networks for object detection,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [94] F. Yang, W. Choi, and Y. Lin, “Exploit all the layers: Fast and accurate cnn object detector with dependent pooling and cascaded rejection classifiers,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2129–2137.
- [95] J. Redmon and A. Farhadi, “Yolo9000: Better, faster, stronger,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [96] T. Kong, F. Sun, A. Yao, H. Liu, M. Lu, and Y. Chen, “Ron: Reverse connection with objectness prior networks for object detection,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [97] Q. Li, S. Jin, and J. Yan, “Mimicking very efficient network for object detection,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [98] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Fu, and A. Berg, “Ssd: Single shot multibox detector,” in *European Conference on Computer Vision (ECCV)*, 2016, pp. 21–37.
- [99] T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, “Focal loss for dense object detection,” in *International Conference on Computer Vision (ICCV)*, 2017, pp. 2999–3007.
- [100] R. Benenson, M. Omran, J. Hosang, and B. Schiele, “Ten years of pedestrian detection, what have we learned?” in *ECCV 2014 Workshop*, 2014, pp. 613–627.
- [101] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? the kitti vision benchmark suite,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012, pp. 3354–3361.
- [102] S. Zhang, R. Benenson, and B. Schiele, “Citypersons: A diverse dataset for pedestrian detection,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

- [103] P. Dollar, Z. Tu, P. Perona, and S. Belongie, “Integral channel features,” in *British Machine Vision Conference (BMVC)*, 2009, pp. 91.1–91.11.
- [104] R. Benenson and M. M., “Seeking the strongest rigid detector,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 3666–3673.
- [105] S. Zhang, C. Bauckhage, and A. Cremers, “Infomed haar-like features improve pedestrian detection,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 947–954.
- [106] S. Paisitkriangkrai, C. Shen, and A. Hengel, “Strengthening the effectiveness of pedestrian detection with spatially pooled features,” in *European Conference on Computer Vision (ECCV)*, 2014, pp. 546–561.
- [107] W. Nam, P. Dollar, and J. Han, “Local decorrelation for improved pedestrian detection,” in *Advances in Neural Information Processing Systems (NIPS)*, 2014, pp. 424–432.
- [108] S. Zhang, R. Benenson, and B. Schiele, “Filtered channel features for pedestrian detection,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1751–1760.
- [109] W. Ouyang and X. Wang, “Joint deep learning for pedestrian detection,” in *International Conference on Computer Vision (ICCV)*, 2013, pp. 2056–2063.
- [110] J. Hosang, M. Omran, R. Benenson, and B. Schiele, “Taking a deeper look at pedestrians,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 4073–4082.
- [111] B. Yang, J. Yan, Z. Lei, and S. Li, “Convolutional channel features,” in *International Conference on Computer Vision (ICCV)*, 2015, pp. 82–90.
- [112] Y. Tian, P. Luo, X. Wang, and X. Tang, “Pedestrian detection aided by deep learning semantic tasks,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 5079–5087.

-
- [113] Z. Cai, M. Saberian, and N. Vasconcelos, “Learning complexity-aware cascades for deep pedestrian detection,” in *International Conference on Computer Vision (ICCV)*, 2015, pp. 3361–3369.
 - [114] L. Zhang, L. Lin, X. Liang, and K. He, “Is faster r-cnn doing well for pedestrian detection?” in *European Conference on Computer Vision (ECCV)*, 2016, pp. 443–457.
 - [115] D. Xu, W. Ouyang, E. Ricci, X. Wang, and N. Sebe, “Learning cross-modal deep representations for robust pedestrian detection,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
 - [116] J. Mao, T. Xiao, Y. Jiang, and Z. Cao, “What can help pedestrian detection?” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
 - [117] S. Hwang, J. Park, N. Kim, Y. Choi, and I. Kweon, “Multispectral pedestrian detection: Benchmark dataset and baseline,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1037–1045.
 - [118] A. Costea and S. Nedeveschi, “Semantic channels for fast pedestrian detection,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2360–2368.
 - [119] Y. Ding and X. J., “Contextual boost for pedestrian detection,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012, pp. 2895–2902.
 - [120] X. Zeng, W. Ouyang, and X. Wang, “Multi-stage contextual deep learning for pedestrian detection,” in *International Conference on Computer Vision (ICCV)*, 2013, pp. 121–128.
 - [121] W. Ouyang, X. Zeng, and X. Wang, “Single-pedestrian detection aided by two-pedestrian detection,” *IEEE Transaction on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 37, no. 9, pp. 1875–1889, 2010.
 - [122] J. Yan, X. Zhang, Z. Lei, S. Liao, and S. Li, “Robust multi-resolution pedestrian detection in traffic scenes,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 3033–3040.

- [123] B. Leibe, E. Seemann, and B. Schiele, “Pedestrian detection in crowded scenes,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005, pp. 878–885.
- [124] V. Shet, J. Neumann, V. Ramesh, and L. Davis, “Bilattice-based logical reasoning for human detection,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007, pp. 1–8.
- [125] B. Wu and R. Nevatia, “Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detectors,” in *International Conference on Computer Vision (ICCV)*, 2005, pp. 90–97.
- [126] M. Fischler and R. Elschlager, “The representation and matching of pictorial structures,” *IEEE Transactions on Computers*, vol. C-22, no. 1, pp. 67–92, 1973.
- [127] Y. Yang and D. Ramanan, “Articulated pose estimation with flexible mixtures-of-parts,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011, pp. 1385–1392.
- [128] W. Yang, W. Y., and G. Mori, “Recognizing human actions from still images with latent poses,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010, pp. 2030–2037.
- [129] L. Zhu, Y. Chen, A. Yuille, and W. Freeman, “Latent hierarchical structural learning for object detection,” in *International Conference on Computer Vision (ICCV)*, 2010, pp. 1062–1069.
- [130] S. Yu and J. Shi, “Multiclass spectral clustering,” in *International Conference on Computer Vision (ICCV)*, 2003, pp. 313–319.
- [131] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes challenge 2010 (voc2010) results,” <http://www.pascal-network.org/challenges/VOC/voc2010/workshop/index.html>, 2010.

-
- [132] R. Girshick, P. Felzenszwalb, and D. McAllester, “Discriminatively trained deformable part models, release 5,” <http://people.cs.uchicago.edu/~rbg/latent-release5/>, 2012.
- [133] D. Hoiem, Y. Chodpathumwan, and Q. Dai, “Diagnosing error in object detectors,” in *European Conference on Computer Vision (ECCV)*, 2012, pp. 340–353.
- [134] G. Ghiasi, Y. Yang, D. Ramanan, and C. Fowlkes, “Parsing occluded people,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 2401–2408.
- [135] G. Nemhauser, L. Wolsey, and M. Fisher, “An analysis of approximations for maximizing submodular set functions,” in *Mathematical Programming*, vol. 14, no. 1, 1978, p. 265–294.
- [136] F. Zhu, Z. Jiang, and L. Shao, “Submodular object recognition,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 2457–2464.
- [137] N. Lazic, I. Givoni, B. Frey, and P. Aarabi, “Floss: Facility location for subspace segmentation,” in *International Conference on Computer Vision (ICCV)*, 2009, pp. 825–832.
- [138] C. Gu and X. Ren, “Discriminative mixture-of-templates for viewpoint classification,” in *European Conference on Computer Vision (ECCV)*, 2010, pp. 408–421.
- [139] J. Zhu, S. Rosset, T. Hastie, and R. Tibshirani, “1-norm support vector machines,” in *Advances in Neural Information Processing Systems (NIPS)*, 2004, pp. 49–56.
- [140] J. Friedman, T. Hastie, and R. Tibshirani, “Additive logistic regression: A statistical view of boosting,” *The Annals of Statistics*, vol. 28, no. 2, pp. 337–407, 2000.
- [141] R. Tibshirani, “Regression shrinkage and selection via the lasso,” *Journal of the Royal Statistical Society, Series B*, vol. 58, pp. 267–288, 1994.
- [142] R. Fan, K. Chang, C. Hsieh, X. Wang, and C. Lin, “Liblinear: A library for large linear classification,” in *Journal of Machine Learning Research*, vol. 9, 2008, pp. 1871–1874.

- [143] R. Flamary, N. Jrad, R. Phlypo, M. Congedo, and A. Rakotomamonjy, “Mixed-norm regularization for brain decoding,” in *Computational and Mathematical Methods in Medicine*, 2014.
- [144] X. Du, M. El-Khamy, J. Lee, and L. S. Davis, “Fused DNN: A deep neural network fusion approach to fast and robust pedestrian detection,” *CoRR*, 2016. [Online]. Available: <http://arxiv.org/abs/1610.03466>
- [145] G. Brazil, X. Yin, and X. Liu, “Illuminating pedestrians via simultaneous detection and segmentation,” in *International Conference on Computer Vision (ICCV)*, 2017.
- [146] S. Wang, J. Cheng, H. Liu, and M. Tang, “Pcn: Part and context information for pedestrian detection with cnns,” in *British Machine Vision Conference (BMVC)*, 2017.
- [147] R. Schapire and Y. Singer, “Improved boosting algorithms using confidence-rated predictions,” *Machine Learning*, vol. 37, no. 3, pp. 297–336, 1999.
- [148] S. Karaoglu, Y. Liu, and T. Gevers, “Detect2rank: Combining object detectors using learning to rank,” *IEEE Transaction on Image Processing (TIP)*, vol. 25, no. 1, pp. 233–248, 2016.
- [149] A. Vedaldi and A. Zisserman, “Efficient additive kernels via explicit feature maps,” *IEEE Transaction on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 34, no. 3, pp. 480–492, 2012.
- [150] C. Toca, M. Ciuc, and C. Patrascu, “Normalized autobinomial markov channels for pedestrian detection,” in *British Machine Vision Conference (BMVC)*, 2015, pp. 175.1–175.13.
- [151] Y. Yang, Z. Wang, and F. Wu, “Exploring prior knowledge for pedestrian detection,” in *British Machine Vision Conference (BMVC)*, 2015, pp. 176.1–176.12.
- [152] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *CoRR*, vol. abs/1409.1556, 2014.